Theoretical Grounds for Popularity Bias in Low-Rank Two-Tower Models Trained with Cosine-Based Loss

Anonymous Authors[1]

Abstract

Low-rank two-tower models may exhibit popularity bias: a positive correlation between item frequency and embedding norm inflates dot-product scores for popular items, making magnitude dominate over directional similarity in the retrieval outcomes. This phenomenon is due to specific properties of loss and encoder architecture. We identify sufficient conditions under which embedding updates are provably orthogonal to the current embedding and hence monotonically increase its norm. Our theory yields explicit guarantees for the emergence of popularity bias in practical two-tower setups (InfoNCE loss, asymmetric two-tower architecture) and corrects a common misconception in prior works that orthogonality of the gradient alone implies norm inflation for deep encoders. Empirical studies support the theory via geometry-first investigation: configurations that satisfy these premises exhibit strictly orthogonal embedding movements and a robust statistically significant frequency–norm coupling, whereas violations of any premise break orthogonality and yield non-systematic update trajectories. The results provide theoretical grounds for popularity bias in cosine-trained two-tower models (particularly in recommender systems, though not limited to them) and show when it should be expected in production systems.

1. Introduction

Two-tower architectures are the de facto standard for large-scale retrieval and recommendation (Huang et al., 2024). Their efficiency stems from decoupling the user and item encoders and scoring by a simple similarity, a principle that has made low-rank matrix factorization one of the most effective approaches in collaborative filtering (Koenigstein et al., 2012), which enables precomputation, approximate nearest-neighbor search, and low-latency serving at web-scale. Yet the role of embedding magnitudes in training and ranking remains under-characterized: empirical observations vary across systems, and theory has not reached consensus on when and why norms change under different objectives and optimizers.

By the definition of cosine similarity, $\cos(q,k) = \langle q,k\rangle/(\|q\|\,\|k\|)$, the normalization appears to cancel the effect of magnitudes, so objectives that couple the left and right encoder embeddings only through cosine similarity are expected to be insensitive to embedding norms. A competing line of analysis argues (Wang et al., 2017; Draganov et al., 2024; 2025) that for cosine-based losses the gradient $\nabla_q \mathcal{L}$ is orthogonal to q, and therefore any gradient step must increase $\|q\|$, often extrapolated to deep encoders. Both perspectives ignore how parameter updates propagate through the encoder and do not explicitly state the algorithmic and architectural conditions under which norm growth necessarily occurs.

We replace black-box reasoning with an update-level analysis. We track how parameter updates map to item-embedding displacements and prove that under explicit and testable conditions each step moves the embedding orthogonally to its current value and thus monotonically increases its squared norm. Building on this, we formalize a frequency–norm mechanism (conditional on per-update increments): popularity increases how often item updates occur, and orthogonality ensures each such update contributes a nonnegative radial increment to $\|q_i\|^2$ (Appendix C).

The analysis also delineates the limits of claims that "gradient orthogonality implies norm inflation" for deep encoders (Wang et al., 2017; Draganov et al., 2024; 2025). Orthogonality of the output gradient is insufficient on its own; orthogonality of the *update* follows only when the encoder's Jacobian preserves it, which holds precisely under the stated premises. Violating any premise — adding nonlinearities or parameter sharing in the item tower or optimizing dot product — breaks the guarantee and leads to non-systematic norm dynamics. Our contributions are:

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

- **Explanation of sufficient conditions for orthogonality of embedding updates.** We prove that embeddings move orthogonally under the following premises: (A1) the item side is optimized by SGD without regularization and momentum; (A2) the item encoder is linear in its parameters; (A3) the item encoder is lookup-equivalent [1] ; and (A4) the loss depends on the item only via cosine. Orthogonality breaks if any premise is violated.

- **Explanation of why orthogonality of embedding updates implies popularity bias.** We prove a formal mechanism (conditional on per-update increments) linking sampling frequency to norm growth and demonstrate that this geometric artifact translates directly into popularity biased retrieval outcomes.

- **Clarification of prior claims in the literature.** We re-examine claims from studies on similar topic (Wang et al., 2017; Draganov et al., 2024; 2025) and argue that gradient update affects the model parameters, not its output itself.

- **The "Baseline Paradox" in architectural design.** We identify a structural paradox where simple baselines, favored for their transparency in early development, are mathematically guaranteed to exhibit bias. This creates a risk of false negatives (discarding viable projects due to optimization artifacts), though complex production setups remain vulnerable too.

Scope and implications. We establish sufficient conditions under which embedding updates are orthogonal and popularity bias emerges. Outside this regime, these effects may or may not occur; we make no claim of necessity. Other sources of popularity bias are examined in prior work (Zhang et al., 2023).

2. Sufficient Conditions for Orthogonality of Embedding Updates

2.1. One Equation that Underpins the Analysis

We consider a canonical two-tower architecture with disjoint parameter sets: a left encoder and a right encoder. Embedding learning in such systems is typically effected by moving the left output closer to or farther from the right output; for example, by objectives that minimize or maximize the dot product between a user embedding and an item embedding (the outputs of the left and right encoders).

A minimal instance is *Alternating Least Squares*, where the two encoders degenerate to the user and item embedding matrices. We, however, develop the analysis for general encoders and introduce architectural restrictions only where they are strictly required by the following mathematics.

Training proceeds by backpropagation (Goodfellow et al., 2016). For a fixed encoder (left or right), we use the following notation:

g_j loss gradient w.r.t. the encoder output q_j for the j-th example in the batch;

J_j Jacobian of q_j w.r.t. all encoder parameters θ.

The relations are

$$g_j = \frac{\partial \mathcal{L}}{\partial q_j},\ J_j = \frac{\partial q_j}{\partial \theta}. \tag{1}$$

It follows that the loss gradient w.r.t. the encoder parameters is

$$g_\theta = \sum_j \frac{\partial q_j}{\partial \theta}\,\frac{\partial \mathcal{L}}{\partial q_j} = \sum_j J_j^\top\, g_j. \tag{2}$$

Therefore, a single SGD step on this batch yields

$$\Delta\theta = -\eta\, g_\theta = -\eta \sum_j J_j^\top\, g_j,\ (\eta \text{ is learning rate}). \tag{3}$$

To relate parameter updates to movements in representation space, we *linearize* the encoder around the current parameters:

$$\boxed{\Delta q_i = J_i\, \Delta\theta = -\eta \sum_j J_i\, J_j^\top\, g_j} \tag{4}$$

This equation serves as the point of departure for the following analysis.

2.2. Interim Focus: Parameter-Linear Encoders

The equation developed above is a *linear approximation* of the embedding dynamics during training. For parameter-linear encoders this linearization is exact: for each example i the map from parameters to the output satisfies $q_i(\theta) = J_i\,\theta$, and hence equation (4) holds without approximation.

For non-linear encoders, the relation $\Delta q_i \approx J_i\,\Delta\theta$ remains accurate under sufficiently small learning rates. We postpone a full treatment of the additional constraints imposed by non-linear architecture to Section 2.4 (*Beyond linear encoders*).

2.3. What Is a Parameter-Linear Encoder

For notation, let $q(\theta, x_i) := q_i$ denote the encoder output with parameters θ given input x_i (e.g., categorical feature or

[1] *Lookup-equivalent* encoder will be defined later in Section 2.4.

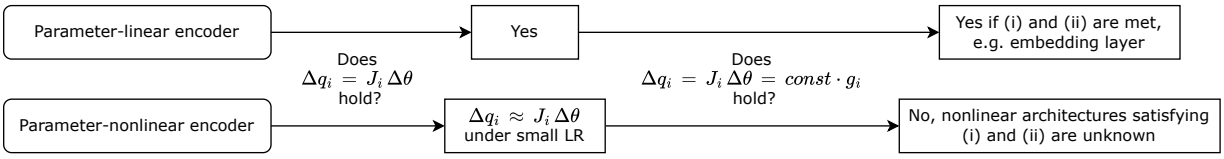

Figure 1. Mind map of Sections 2.1–2.4. Collinearity $\Delta q_i \parallel g_i$ holds only for a single embedding layer and a single linear layer (if the distinct inputs are pairwise orthogonal).

feature vector for example i), and $J(x_i) := J_i$ the Jacobian $\partial q(\theta, x_i)/\partial\theta$.

The encoder is *parameter-linear* if its output for any input example i can be written in the exact linear form

$$q(\theta, x_i) = J(x_i)\,\theta \tag{5}$$

where $J(x_i)$ does not depend on θ and is therefore constant over the parameter space.

Consequently, if the encoder is parameter-linear, then equation (4) holds without approximation.

Examples are deferred to Appendix A: a single embedding layer (Appendix A.1) and a single linear layer without bias (Appendix A.2) are parameter-linear architectures.

By contrast, encoder which consists of two consecutive linear layers without bias is the example of an architecture which is not parameter-linear (Appendix A.3).

2.4. For Some Linear Encoders, the Embedding Update Is Collinear with the Loss Gradient

Drawing on equation (4), we now characterize when the embedding displacement is collinear with the output gradient:

$$\Delta q_i \;=\; J_i\,\Delta\theta \;=\; -\eta \sum_j J_i\,J_j^\top\,g_j \tag{6}$$

Collinearity of the loss gradient $g_i = \nabla_{q_i}\mathcal{L}$ with the embedding update Δq_i arises when the pairwise Jacobian products satisfy the following operator conditions:

- **(i)** $J_i J_j^\top = 0$ for $x_j \neq x_i$ (all non-i terms vanish in the batch sum, thus other gradients g_j do not affect Δq_i);
- **(ii)** $J_i J_i^\top = \alpha_i I_d$ for some scalar α_i ($J_i J_i^\top g_i = \alpha_i g_i$, i.e., the operator rescales but does not rotate g_i).

Under (i)–(ii), if x_i occurs only once in the batch, the sum in (6) reduces to

$$\Delta q_i \;=\; -\eta\,\alpha_i\,g_i, \tag{7}$$

that is, the update is a scalar multiple of g_i. We call an encoder *lookup-equivalent* if it is (a) a single embedding layer, or (b) a single bias-free linear layer applied to one-hot inputs. For these two cases, the Jacobian products satisfy (i)–(ii), as demonstrated in the following paragraph.

For a single embedding layer (Appendix A.1),

$$J_i J_j^\top \;=\; \begin{cases} I_d, & x_i = x_j, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

If c_i denotes the number of occurrences of x_i in the batch, substituting (8) into (6) yields

$$\Delta q_i \;=\; -\eta \sum_{j:\,x_j = x_i} g_j \;=\; -\eta\,c_i\,\bar{g}_i, \tag{9}$$

where $\bar{g}_i := \frac{1}{c_i}\sum_{j:\,x_j=x_i} g_j$. Hence the update is strictly collinear with the (batch-aggregated) gradient direction $\bar{g}_i$.

For a single linear layer (Appendix A.2),

$$J_i J_j^\top \;=\; (x_i^\top x_j)\,I_d. \tag{10}$$

Therefore,

$$\Delta q_i \;=\; -\eta\,\|x_i\|^2 \sum_{j:\,x_j = x_i} g_j \;-\; \eta \sum_{j:\,x_j \neq x_i} (x_i^\top x_j)\,g_j \tag{11}$$

If the distinct inputs are pairwise orthogonal (e.g., one-hot features or, more generally, $x_i^\top x_j = 0$ for $x_i \neq x_j$), the cross-terms vanish and self-terms aggregate over the c_i occurrences of x_i in the batch,

$$\Delta q_i \;=\; -\eta\,\|x_i\|^2 \sum_{j:\,x_j = x_i} g_j \;=\; -\eta\,c_i\,\|x_i\|^2\,\bar{g}_i, \tag{12}$$

which is again collinear with $\bar{g}_i$.

These cases illustrate a general principle: whenever $J_i J_j^\top$ is diagonal in the sense of $J_i J_j^\top = \alpha_{ij} I_d$ with $\alpha_{ij} = 0$ for $x_j \neq x_i$, the embedding update aligns with the output gradient. In particular, with one-hot features (orthogonal inputs), a bias-free linear layer and an embedding lookup are equivalent up to implementation (MatMul vs. lookup).

Beyond linear encoders. For encoders which are not parameter-linear, we are not aware of natural conditions under which (i) and (ii) hold, even when the first-order approximation $\Delta q_i \approx J_i\,\Delta\theta$ is accurate (e.g., under small learning rates). In practice, non-linearity perturbs J_i in an input- and parameter-dependent way that breaks both the cross-term annihilation (i) and the isotropy (ii). See Figure 1 for a mind map of conditions for collinearity.

2.5. For Any Cosine-Based Loss, the Output Gradient Is Orthogonal to the Encoder Output

Let the embedding q produced by the encoder participate in the loss only via cosine similarities with other embeddings generated by the two-tower model. Then the loss can be written as

$$\mathcal{L} = F\big(\cos(q, k_1), \dots, \cos(q, k_r)\big). \tag{13}$$

And its gradient with respect to q is

$$\nabla_q \mathcal{L} = \sum_{s=1}^{r} \frac{\partial F}{\partial \cos(q, k_s)} \nabla_q \cos(q, k_s). \tag{14}$$

Here the derivative of the loss with respect to a cosine is a scalar. We can say that the loss gradient with respect to q is orthogonal to q if

$$\langle q, \nabla_q \mathcal{L} \rangle = 0. \tag{15}$$

$$\langle q, \nabla_q \mathcal{L} \rangle = \sum_{s=1}^{r} \frac{\partial F}{\partial \cos(q, k_s)} \langle q, \nabla_q \cos(q, k_s) \rangle = 0, \tag{16}$$

here we used lemma B.1, which states that $\langle q, \nabla_q \cos(q, *) \rangle = 0$. Hence $q_i \perp g_i$ for any cosine loss. Consequently (for duplicates use $\bar{g}_i$),

$$\textit{if } \Delta q_i \parallel g_i, \textit{ then } q_i \perp \Delta q_i, \tag{17}$$

which implies that during training the embedding moves orthogonally (see Figure 2).

2.6. Theorem: When Embeddings Move Orthogonally

Theorem 2.1 (Orthogonal Embedding Motion)**.** *If the encoder satisfies the following four conditions simultaneously:*

1. *optimization uses SGD without momentum*[2] *and without regularization (the only way to guarantee strict movement defined by equations),*
2. *the encoder is linear in its parameters (parameter-linear),*
3. *the encoder is lookup-equivalent*[3]*,*
4. *the loss gradient with respect to the encoder output (the embedding) is orthogonal to that output (e.g., cosine-based loss),*

then, during training, the embedding moves orthogonally, i.e., tangentially to hypersphere on which it was located at the previous step.

Proof. SGD without momentum gives $\Delta\theta = -\eta\, g_\theta$. Parameter-linearity makes (4) exact. Lookup-equivalence yields $\Delta q_i \parallel g_i$ (for duplicates use $\bar{g}_i$; Section 2.4). Condition 4 gives $q_i \perp g_i$ (for duplicates use $\bar{g}_i$; Section 2.5). Hence $q_i \perp \Delta q_i$ by (17). □

2.7. Related work on orthogonal movement

In (Wang et al., 2017) a new way to train deep encoders is proposed. In Section 3.2 they prove that, at the model output, for x and the gradient $\partial\mathcal{L}/\partial x$, orthogonality holds, after which they immediately make the logical transition

> *"it can be inferred that after update, $\|x\|^2$ always increases"* (Wang et al., 2017, p. 4).

That is, they implicitly assume that if the loss gradient with respect to the model output is orthogonal to the model output, then this output itself also moves orthogonally. This transition is not obvious: it switches from orthogonality of the gradient to orthogonality of the embedding displacement itself.

Such orthogonality of the step is guaranteed only under a sufficiently strict set of conditions. We write these conditions explicitly and show that, when they are satisfied, embeddings do move orthogonally; outside this region the transition from orthogonality of the gradient to orthogonality of the embedding displacement (and hence to monotone growth of the embedding norm) may fail. Consequently, the motivation for modifying training *because the update increases the point's norm* should be treated as a qualified claim: it is true in the configurations described by our conditions, but it is not a universal basis for changing how encoders are trained—especially deeper ones.

The same critique applies to (Draganov et al., 2024; 2025). In particular, the former states:

> *"The gradient is orthogonal to the embedding, so a step of gradient descent can only increase the embedding's magnitude"* (Draganov et al., 2024, p. 10),

and also claims

> *"these derivations are extremely general—they hold across deep learning architectures"* (Draganov et al., 2024, p. 1),

without accounting for the fact that the gradient is taken with respect to the model output rather than the parameters;

[2] Momentum accumulates past gradients, so even if Conditions 2–4 hold, the update need not remain orthogonal to the current embedding.

[3] This does imply parameter-linearity, but we retain both - one makes the linearization exact, other ensures collinearity.

it also supports the claim only by observing norm growth in experiments, without analyzing whether that growth is indeed caused by orthogonal displacement of the embedding.

The latter states:

> *"Because gradient updates are orthogonal to an embedding point, they must increase the point's norm"* (Draganov et al., 2025, p. 2),

which is correct only after refining the conditions under which the embedding displacement (and not just the gradient) is orthogonal. Such a simplification can mislead the reader about the universality of the "inevitable catch-22".

It should be acknowledged that in Appendix B they provide a qualification, stating

> *"We optimize the cosine similarity by performing standard gradient descent on the embeddings themselves"* (Draganov et al., 2025, p. 17),

which implies a shallow architecture such as an Embedding Layer; however, this architectural restriction (if intended) is fixed only in the appendix, while the main text presents the correctness of the previous claim without specifying the architecture.

When the conditions above do hold, our analytic proof of orthogonal embedding displacement indeed implies growth of the embedding norm (by the Pythagorean theorem). A natural question then arises: how should this growth be evaluated analytically?

3. Embedding-Norm Is Monotonic in Item Popularity

Throughout this section (as well as corresponding appendices) we assume the four conditions from Section 2 hold (SGD without momentum/regularization; parameter-linear encoder; lookup-equivalent encoder; cosine-loss gradient orthogonal to the output). Under these assumptions the embedding moves orthogonally at each update. By the Pythagorean theorem,

$$\left\|q_i^{(t+1)}\right\|^2 - \left\|q_i^{(t)}\right\|^2 \;=\; \left\|\Delta q_i^{(t)}\right\|^2. \tag{18}$$

Finally, we analyze the dependence of the embedding-norm growth on item popularity. We assume standard uniform-over-interactions batch sampling, under which the per-slot probability p_i is proportional to item frequency (popularity) in the training data; other schemes (uniform-over-items, hard negatives) are deliberate modifications of this default. Appendix C formalizes a coupling *mechanism* that isolates the effect of sampling probability and proves that, conditional on per-update increments, the expected *squared* norm after T batches is nondecreasing in the item's sampling probability; extending this to a full SGD statement would require separate control of trajectory-dependent update magnitudes, which lies outside the scope of the present paper.

Takeaway. This section and Theorem C.2 provide a formal mechanism for popularity bias: under the four assumptions of Section 2, each update contributes a nonnegative radial increment to $\|q_i\|^2$, and more frequent sampling increases how often those increments are accumulated. To observe a statistically significant popularity bias in practice, an additional (fifth) factor is required; it will be discussed in the next section.

4. Experiments: Empirical Validation of Theoretical Results

To clarify scope, we do not assess recommendation quality (e.g., MAP@k), given that popularity bias can be either harmful or beneficial depending on the application context (Klimashevskaia et al., 2024). Such evaluation lies outside the contribution of this work. Instead, we empirically examine the full causal chain predicted by the theory: from the orthogonality of updates and frequency–norm coupling (geometric properties) to the resulting skewed retrieval outcomes. Datasets and implementation details are available in a public GitHub repository.

4.1. Under Which Factors Orthogonality of Embedding Movement Emerges

See Table 1. In this subsection we use a small toy synthetic dataset in which personalized dependencies can be observed visually.

Definitions:
Condition 1: the encoder architecture consists of either (i) a single linear layer applied to one-hot inputs, or (ii) a single embedding layer.
Condition 2: the gradient of the loss with respect to the encoder output is orthogonal to that output.

Clarifications. In all experiments the right tower (item tower) is a simple embedding layer (nn.Embedding); InfoNCE (van den Oord et al., 2018) denotes the contrastive softmax loss with in-batch negatives, where "cos" (resp., "dot") refers to cosine (resp., dot-product) similarity. We use SGD without momentum and without regularization; alternative optimizers did not reveal any movement patterns, although we will separately consider mixed-optimizer settings, including Adam, later (see Experiment 10). In particular, SGD with momentum under the remaining conditions (Theorem 2.1) breaks orthogonality; this Experiment 0 is included in the repository (GitHub) among all experiments but omitted from the table as a trivial consequence. In Ex-

Table 1. First subsection of experiments — orthogonality of embedding movement across left (user) encoder architectures and losses. See definitions on "Cond. 1" and "Cond. 2" in subsection definitions.

No.	Left (user) encoder	Loss	Cond. 1 for left encoder?	Cond. 2?	Orthogonal trajectory?
1	Embedding	InfoNCE (cos)	✓	✓	✓
2	Embedding	InfoNCE (**dot**)	✓	×	×
3	BERT-like	InfoNCE (cos)	×	✓	User × / Item ✓
4	Embedding → Linear	InfoNCE (cos)	×	✓	User × / Item ✓
5	Embedding (frozen) → Linear	InfoNCE (cos)	×	✓	User × / Item ✓
6	one-hot → Linear	InfoNCE (cos)	✓	✓	✓
7	Embedding	$\big(1-\cos(q,k)\big)^2$	✓	✓	✓

periment 4 the architecture coincides with one-hot → Linear → Linear, i.e., simply two linear layers. The architecture in Experiment 5 is an example of an encoder that consists of a single Linear layer without the input-orthogonality requirement (no one-hot inputs): the inputs are just some numeric features.

Experiments 1 and 6: orthogonal movement confirmed. Both setups satisfy all four sufficient conditions; every embedding displacement lands on a strictly larger concentric circle (Figure 2), directly reflecting the Pythagorean norm increment of Eq. (18). The same orthogonal pattern is observed for both the user and item towers in these two experiments.

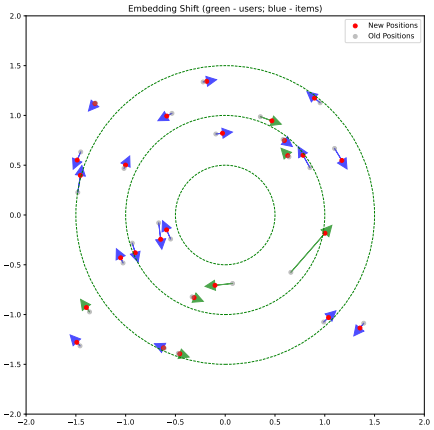

Figure 2. Experiments 1 and 6. Orthogonal movement during training — the displacement vectors (arrows) are tangent to the circles.

If the encoder is nonlinear in the parameters: Recall that two linear layers constitute nonlinearity with respect to the model parameters. We argued that due to this nonlinearity one cannot obtain an analytical explanation of how exactly embeddings move. The result of Experiment 4 is consistent with this reasoning: unlike Experiments 1 and 6, here the embeddings from the left encoder move chaotically (see Figure 3).

If, in a parameter-linear encoder, the inputs are not mutually orthogonal: Recall that linearity of the encoder is still insufficient to guarantee orthogonal movement of embeddings, and consider the result of Experiment 5: embeddings produced by the left encoder move chaotically (see Figure 3 — same as Experiment 4). This agrees with our statement that, for orthogonality of embedding movement, each unique input to the encoder must have its own parameter row that does not intersect with others. We were able to identify only two (essentially identical) architectures where this holds — a linear layer over one-hot vectors (Experiment 6) or an embedding layer (Experiment 1).

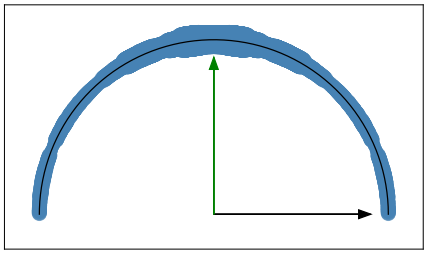

Figure 3. Experiments 4 and 5. User-update angles distribution on semicircle; black arrow is user embedding, green arrow is orthogonal direction.

If a different loss is used: The result of Experiment 7 does not contradict our statement that the gradient of any cosine-based loss with respect to the encoder output is orthogonal to that output.

Overall, these experiments confirm that strict orthogonality requires precise conditions: violations lead to non-systematic, chaotic movement. For example, the angle distribution for Experiments 4 and 5 (Figure 3) shows no strong concentration around 90 (share within ± 10: $\approx 20\%$) and spans the full range $[0, 180]$.

4.2. Does Orthogonality of Embedding Movement Imply Popularity Bias?

See Table 2. In this subsection we use MovieLens 32M dataset (Harper & Konstan, 2015) and the same InfoNCE cosine-based loss.

In Experiments 8 and 9 we test whether the orthogonality

Table 2. Last two subsections of experiments — the emergence and detectability of popularity bias. Exp. 8 and 9 report whether displacement is sufficient and quantify popularity bias both as a geometric frequency–norm correlation and as a retrieval skew. The retrieval bias is measured as the share of popular items (from the top-1% by training frequency) appearing in the top-100 recommendations. Exp. 10 demonstrates that this mechanism persists in a production-like asymmetric setup.

No.	Left (user) encoder	Cond. 1 for left encoder?	Cond. 2?	Orthogonal trajectory?	Suff. orthogonal displacement?	Item Pop–Norm Correlation?	Top-1% (dot)	Top-1% (cos)
8	Embedding	✓	✓	✓	✓ (high LR)	✓	**0.72**	0.29
9	Embedding	✓	✓	✓	× (low LR)	×	0.007	0.007
10	BERT-like (with Adam optimizer)	×	✓	User × Item ✓	✓ (high LR)	✓	**0.63**	0.32

property leads to the emergence of popularity bias.

Result. If embeddings move orthogonally under the factors described above, they do increase their norms; thus items that "receive movement" more often (more popular items, appearing more frequently in training batches) systematically accumulate larger norms. We observe a substantial and statistically significant correlation between item popularity and embedding norm (Pearson correlation 0.66; the null hypothesis of zero correlation is rejected at $p < 0.05$). However, there is an important caveat, discussed next.

Nuance. We identify an important factor required for the appearance of popularity bias under orthogonal movement: embeddings must move with sufficiently large orthogonal displacement. Purely tangential movement over very short distances yields almost no increase in the embedding norm (by the Pythagorean theorem) and cannot overcome the random norm at initialization. In Experiment 8 we increased the learning rate of the SGD optimizer so that objects moved over larger distances; with a commonly used smaller learning rate (LR $= 0.1$) objects moved extremely slowly and effectively did not leave their previous-radius hyperspheres, and we did not observe a statistically significant popularity–norm correlation (Experiment 9).

This finding is consistent with the analysis from Section 3.

The last two columns of Table 2 confirm that this geometric phenomenon translates directly into retrieval outcomes. In Experiment 8, where norms grow orthogonally, the share of popular items in the top-100 results is drastically higher for dot-product ranking (72%) compared to cosine (29%). Conversely, in Experiment 9, where norms barely change, both ranking methods yield identical distributions (≈0.7%).

4.3. Popularity Bias in Practical Two-Tower Setup

Here we consider an asymmetric two-tower model that is close to a production training design (Experiment 10, Table 2, same MovieLens 32M dataset (Harper & Konstan, 2015) and InfoNCE cosine-based loss): the left tower is a deep BERT over the user's history (Sun et al., 2019); the right tower is, again, a simple embedding layer over the item identifier. Using such a simple architecture for item tower is a popular scenario when representing an item without convolutions and content features is an intentional design choice. For instance, such an approach was adopted for YouTube (Covington et al., 2016) by using a trainable item (class) embedding as the input to a softmax loss; (Yuan et al., 2024) likewise employs a simple item-embedding matrix as an item tower, arguing that a complex item-side architecture may not provide substantial gains:

> *"Shallow embedding matrices are very effective"* (Yuan et al., 2024, p. 6).

We use separate optimizers for the two towers: user BERT is trained by Adam with weight decay, and the item embedding layer is trained by SGD without momentum and without regularization. In both cases we use a Cyclic LR schedule (Smith, 2017).

Empirical results are consistent with the theory: when the right (item) tower and its training dynamics remain simple — note that a dynamic learning-rate schedule does not invalidate the orthogonal- movement result — item embeddings continue to move orthogonally even in the presence of a complex left-tower architecture. Given sufficiently large orthogonal displacement (e.g., LR=10 for item tower), a statistically significant popularity bias emerges, observed as a positive correlation between an item's embedding norm and its frequency in the training set (Pearson correlation 0.56; the null hypothesis of zero correlation is rejected at $p < 0.05$).

Furthermore, this norm inflation directly skews retrieval results. As shown in Table 2 (Experiment 10), the share of popular items in the top-100 recommendations is nearly doubled when ranking by dot product (63%) compared to cosine similarity (32%). This gap confirms that the learned norms act as a popularity prior, dominating the directional similarity. Figure 4 illustrates the dynamics of this effect during training: while cosine-based retrieval acts as a magnitude-agnostic baseline, dot-product retrieval becomes progressively more

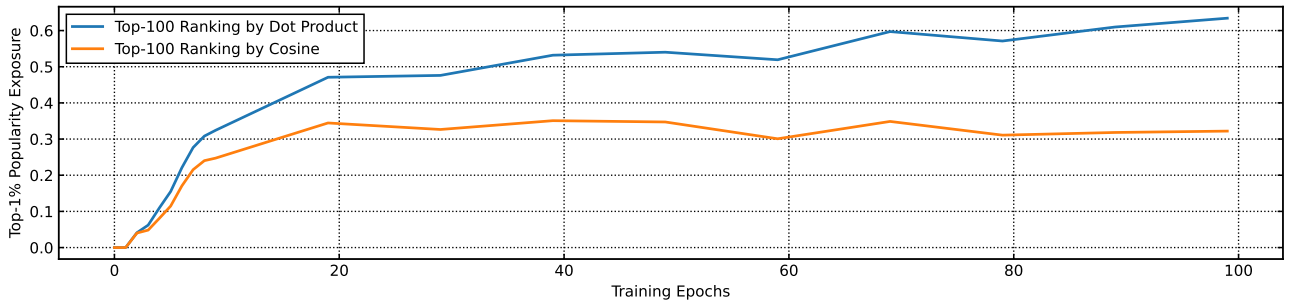

Figure 4. Dynamics of popularity bias in a production-like asymmetric setup (Experiment 10: BERT User Tower + Simple Item Tower). As training progresses, the item embedding norms grow (due to orthogonal updates), causing the share of popular items in dot-product retrieval to diverge sharply from the cosine baseline.

biased as embedding norms grow orthogonally.

5. Conclusion

If the following five factors hold simultaneously for the item tower, then item embeddings move orthogonally, their norms grow monotonically with updates, and a frequency–norm mechanism (conditional on per-update increments) for popularity bias is induced (Appendix C):

1. optimization uses SGD without momentum and without regularization (the only way to guarantee strict movement defined by equations),
2. the encoder is linear in its parameters (parameter-linear),
3. the encoder is lookup-equivalent,
4. the loss gradient with respect to the encoder output (the embedding) is orthogonal to that output (e.g., cosine-based loss),
5. the learning rate is sufficiently large (orthogonality itself does not require this; the point is to overcome the random initialization of embeddings by ensuring sufficiently large orthogonal displacement).

This list makes explicit how many requirements must be met so that embeddings move orthogonally, their norms grow monotonically, and this growth yields a popularity-bias mechanism. Our investigation thereby refines the takeaways of prior work by specifying exactly which factors guarantee orthogonal embedding movement. To our knowledge, these aspects have not been articulated in the literature on representation learning with cosine–based objectives.

An important observation is that, although these strict conditions restrict the encoder architecture for which we can provide guarantees of popularity bias, modern low-rank two–tower systems contain two independent encoders. Even if one is a complex deep model, the other — provided its architecture is sufficiently primitive — already guarantees this phenomenon when trained with a cosine loss.

Finally, our theoretical characterization revisits the academic stance on embedding geometry, offering a structural perspective on architectural design: since the identified sufficient conditions serve as the mechanism for frequency–norm coupling, violating any of them breaks the mathematical guarantee of this bias. Yet, widespread designs employing simple item towers remain fully susceptible to it, which may be critical for downstream ranking tasks — such as dot-product index search — unless this property of the training process is consciously intended as desirable.

Paradoxically, this implies that simpler "baseline" architectures are often the most vulnerable to this phenomenon, as they perfectly satisfy these conditions. This is particularly relevant for the iterative development of recommender systems, where practitioners typically begin with such clean baselines to validate pipelines. Our analysis suggests a hidden pitfall: these baselines are uniquely predisposed to maximal popularity bias due to their geometric transparency. This creates a risk of false negative evaluations early in the development cycle, where a project might be discarded for "learning only popularity", when in fact this behavior is an optimization artifact of baseline simplicity rather than a fundamental limitation of the data.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

Draganov, A., Vadgama, S., and Bekkers, E. The hidden pitfalls of the cosine similarity loss. *arXiv:2406.16468*, 2024.

Draganov, A., Vadgama, S., Damrich, S., B"ohm, J. N., Maes, L., Kobak, D., and Bekkers, E. On the importance of embedding norms in self-supervised learning. *arXiv:2502.09252*, 2025.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

Harper, F. M. and Konstan, J. A. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19, 2015. doi: 10.1145/2827872.

Huang, J., Chen, J., Lin, J., Qin, J., Feng, Z., Zhang, W., and Yu, Y. A comprehensive survey on retrieval methods in recommender systems. *ACM Transactions on Information Systems*, 2024.

Klimashevskaia, A., Jannach, D., Elahi, M., and Trattner, C. A survey on popularity bias in recommender systems. *User Modeling and User-Adapted Interaction*, 2024.

Koenigstein, N., Ram, P., and Shavitt, Y. Efficient retrieval of recommendations in a matrix factorization framework. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pp. 535–544, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311564. doi: 10.1145/2396761.2396831. URL https://doi.org/10.1145/2396761.2396831.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM '19*, pp. 1441–1450, 2019.

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

Wang, F., Xiang, X., Cheng, J., and Yuille, A. L. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM International Conference on Multimedia*, 2017.

Yuan, Y., Zhang, Z., He, X., Nitta, A., Hu, W., Wang, D., Shah, M., Huang, S., Stojanović, B., and Krumholz, A. Contextgnn: Beyond two-tower recommendation systems. *arXiv:2411.19513*, 2024.

Zhang, Y., Zhu, H., Song, Z., Koniusz, P., and King, I. Mitigating the popularity bias of graph collaborative filtering: A dimensional collapse perspective. In *Advances in Neural Information Processing Systems*, 2023.

A. Encoder Examples: Parameter-Linear and Nonlinear

A.1. Encoder with a Single Embedding Layer

Input: $x_i = i \in \{1, \dots, N\}$ (x_i – index of example i).

Parameters: $E \in \mathbb{R}^{N\times d}$, $\theta = E$, $\theta_{i,n} = E_{i,n}$ (n – column index).

Output: $q(\theta, x_i) = E_i \in \mathbb{R}^d$, $q_i = q(\theta, x_i)$.

Let m denote an output coordinate of q, and let i' and n' be arbitrary row/column indices with respect to which the derivative is taken (they range over all parameters). We also vectorize θ from an $N \times d$ matrix into a vector of length $N \cdot d$.

Jacobian:

$$J_i = \frac{\partial q_i}{\partial \theta} = \begin{cases} 1, & \text{if } i' = i,\ n' = m, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Here J_i is a $d \times (N \cdot d)$ table that shows how each output coordinate changes under an infinitesimal change of each parameter at fixed input x_i.

Equivalently, all entries are zero except the $d \times d$ block corresponding to row i, which is the identity I_d:

$$J_i = \Big[\underbrace{0_{d\times(i-1)d}}_{\text{params } 1\dots(i-1)} \Big| \underbrace{I_d}_{\text{params of } i} \Big| \underbrace{0_{d\times(N-i)d}}_{\text{params } (i+1)\dots N} \Big]. \tag{20}$$

Thus J does not depend on θ, and the encoder is parameter-linear.

A.2. Encoder with a Single Linear Layer (No Bias)

Input: $x_i = (x_{i,1}, \dots, x_{i,h})^\top \in \mathbb{R}^h$ (x_i – feature vector of example i).

Parameters: $W \in \mathbb{R}^{d\times h}$, $\theta = W$, $\theta_{m,n} = W_{m,n}$ (here m indexes output rows and n indexes input features).

Output: $q(\theta, x_i) = W\, x_i \in \mathbb{R}^d$, $q_i = q(\theta, x_i)$.

Let m denote an output coordinate of q_i, and let m' and n' be arbitrary row/column indices with respect to which the derivative is taken (they range over all parameters). We also vectorize θ from a $d\times h$ matrix into a vector of length $d\cdot h$.

Jacobian:

$$J_i = \frac{\partial q_i}{\partial \theta} = \begin{cases} x_{i,n'}, & \text{if } m' = m, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Here J_i is a $d \times (d \cdot h)$ table showing how each output coordinate changes under an infinitesimal change of each parameter at fixed input x_i. Equivalently, in each of the d blocks (width h) on row m there is a copy of $x_i^\top$, the remaining entries are zero:

$$J_i = \operatorname{diag}\big(\underbrace{x_i^\top, x_i^\top, \ldots, x_i^\top}_{d \text{ times}}\big). \tag{22}$$

Hence J depends on the input x_i but not on θ, so this encoder is parameter-linear. If a bias is present, append an identity block on the right.

A.3. Encoder with Two Consecutive Linear Layers (No Bias)

Input: $x_i = (x_{i,1}, \ldots, x_{i,h})^\top \in \mathbb{R}^h$ (x_i – feature vector of example i).

Parameters: $A \in \mathbb{R}^{d_1 \times h}$, $B \in \mathbb{R}^{d_2 \times d_1}$, $\theta = \{A, B\}$, $\theta^{(A)}_{r,s} = A_{r,s}$, $\theta^{(B)}_{m,n} = B_{m,n}$ (here r is a "hidden" coordinate, m an output coordinate).

Output: $z_i = A\, x_i \in \mathbb{R}^{d_1}$, $q_i = q(\theta, x_i) = B\, z_i \in \mathbb{R}^{d_2}$.

Let m denote an output coordinate of q_i. Let m', n' be arbitrary row/column indices in B, and r', s' indices in A.

W.r.t. B:

$$\frac{\partial q_{i,m}}{\partial B_{m',n'}} = \begin{cases} z_{i,n'}, & m' = m, \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

W.r.t. A:

$$\frac{\partial q_{i,m}}{\partial A_{r',s'}} = B_{m,r'}\, x_{i,s'}. \tag{24}$$

Therefore, $J_i = \big[\, J_i^{(B)} \;\big|\; J_i^{(A)} \big]$, where $J_i^{(B)} \in \mathbb{R}^{d_2 \times (d_2 d_1)}$ has, in each of the d_2 blocks of width d_1, the row $z_i^\top$, and $J_i^{(A)} \in \mathbb{R}^{d_2 \times (d_1 h)}$ consists of $d_1 h$ columns of the form $B_{:,r'}\, x_{i,s'}$.

Consequently, J_i depends on the parameters (through B), so this encoder is *not* parameter-linear. Intuitively, although the encoder is linear in the *input*, changing a parameter in the first linear layer affects the output through a multiplication by the second layer, which makes the Jacobian parameter-dependent; hence strict parameter-linearity fails.

B. Cosine Orthogonality Lemma

Lemma B.1. *For any nonzero $q, k \in \mathbb{R}^d$, $q^\top \nabla_q \cos(q, k) = 0$.*

Proof. Let

$$\cos(q, k) = f(q) = \frac{u(q)}{v(q)}, \qquad u(q) = q^\top k,\ v(q) = \|q\|\,\|k\|. \tag{25}$$

By the quotient rule,

$$\nabla_q f = \frac{v\, \nabla_q u - u\, \nabla_q v}{v^2}. \tag{26}$$

The terms are $\nabla_q u = k, \nabla_q v = \frac{\|k\|}{\|q\|}\, q$. Substituting and simplifying,

$$\nabla_q f = \frac{\|q\|\,\|k\|\, k - (q^\top k)\, \frac{\|k\|}{\|q\|}\, q}{\big(\|q\|\,\|k\|\big)^2} = \frac{1}{\|q\|\,\|k\|}\Big(k - \frac{q^\top k}{\|q\|^2}\, q\Big). \tag{27}$$

Thus

$$\begin{aligned} q^\top \nabla_q f &= \frac{1}{\|q\|\,\|k\|}\Big(q^\top k - \frac{q^\top k}{\|q\|^2}\, q^\top q\Big) \\ &= \frac{1}{\|q\|\,\|k\|}\Big(q^\top k - \frac{q^\top k}{\|q\|^2}\, \|q\|^2\Big) = 0. \qquad \square \end{aligned} \tag{28}$$

C. Popularity Dependence via Coupling

Because the four conditions of Section 2 alone do not control how update magnitudes vary with batch composition, the result below is stated as a *mechanism* that isolates the contribution of sampling frequency.

Definition C.1 (Conditional-on-increments model)**.** *Fix an item identity i. Consider any training run and let N_T be the number of* update events *for item i up to time T. For each update event $k \in \{1, \ldots, N_T\}$ define $\delta_k := \|\Delta q_i\|^2$, the squared displacement of the item embedding at that update. The* conditional-on-increments model *treats the nonnegative sequence $\{\delta_k\}_{k \geq 1}$ as fixed (non-random) and compares only the effect of varying the sampling frequency (hence the update count N_T). We do* not *claim that $\{\delta_k\}$ is invariant under changing the training sampling policy in full SGD dynamics.*

Step 1: what Section 2 gives (monotone growth per update). Let $q_i^{(t)}$ denote the item embedding after batch t and $s_t := \|q_i^{(t)}\|^2$. Under the four conditions of Section 2, the item embedding update is orthogonal to the current embedding, so by the Pythagorean theorem each update satisfies

$$s_{t+1} = s_t + \|\Delta q_i^{(t)}\|^2, \qquad \|\Delta q_i^{(t)}\|^2 \geq 0. \tag{29}$$

Thus s_t is nondecreasing along training whenever item i is updated.

Step 2: what sampling gives (more popular ⇒ more updates). Assume batches are formed by i.i.d. sampling *with replacement* from a fixed distribution (independent slots within a batch and independent batches over time), and item i has per-slot probability p_i (proportional to its frequency under standard sampling, as assumed in Section 3). With batch size B, the probability that item i appears at least once in a batch is

$$\pi(p_i) := 1 - (1 - p_i)^B. \tag{30}$$

Let $N_T^{(p)}$ be the number of batches among $t = 1, \dots, T$ in which item i appears at least once. Then

$$N_T^{(p)} \sim \mathrm{Binomial}\big(T, \pi(p)\big), \tag{31}$$

and therefore for $p_i' < p_i''$ we can couple the two binomials so that $N_T^{(p_i'')} \geq N_T^{(p_i')}$ almost surely (standard Bernoulli coupling via shared uniforms).

Step 3: a clean conditional-on-increments model that isolates popularity. Here an *update event* refers to a batch in which item i appears at least once and therefore receives an update from that batch's SGD step. A batch may contain multiple occurrences (duplicates) of item i; their combined effect is captured by the single-step squared displacement $\delta_k = \|\Delta q_i\|^2$ for that batch, so we do not assume "one occurrence per batch".

The squared-norm process can always be written as a sum of nonnegative increments along the update events of item i. Define $\delta_k \geq 0$ as the squared displacement at the k-th update of item i (i.e., $\delta_k := \|\Delta q_i\|^2$ at that update). Then, for any horizon T,

$$\|q_i^{(T)}\|^2 = \|q_i^{(0)}\|^2 + \sum_{k=1}^{N_T^{(p_i)}} \delta_k. \tag{32}$$

In a *conditional-on-increments model*, we treat the nonnegative sequence $\{\delta_k\}_{k\geq 1}$ as fixed and compare only the effect of changing the sampling probability p_i (hence changing the random update count $N_T^{(p_i)}$). Let $f(n) := \|q_i^{(0)}\|^2 + \sum_{k=1}^{n} \delta_k$. Since $\delta_k \geq 0$, f is nondecreasing in n. Under the monotone coupling of $N_T^{(p)}$ above, we obtain the *pathwise* inequality

$$f\Big(N_T^{(p_i'')}\Big) \geq f\Big(N_T^{(p_i')}\Big) \quad \text{almost surely, and therefore} \tag{33}$$

$$\mathbb{E}\big[\|q_i^{(T)}\|^2\big]_{p_i''} \;\geq\; \mathbb{E}\big[\|q_i^{(T)}\|^2\big]_{p_i'}. \tag{34}$$

Theorem C.2 (Popularity–Norm Coupling)**.** *Assume the four conditions of Section 2 hold (SGD without momentum / regularization; parameter-linear encoder; lookup-equivalent encoder; the loss gradient with respect to the encoder output is orthogonal to it) and sampling assumptions of Step 2. Fix an item identity i, and in the conditional-on-increments model treat the nonnegative sequence of per-update squared displacements $\{\delta_k\}_{k\geq 1}$ as fixed. Then, for every training horizon T and every pair of sampling probabilities $p_i' < p_i''$,*

$$\mathbb{E}\big[\|q_i^{(T)}\|^2\big]_{p_i''} \;\geq\; \mathbb{E}\big[\|q_i^{(T)}\|^2\big]_{p_i'}.$$

Proof. Steps 1–3 above establish the result. □