Magnitude Attention: Don't Let Group of Similar Keys Steal Your Probability Mass

Vuk Rosić

February 15, 2026

Attention is Blind to Geometry

Standard attention calculates weights based purely on pairwise compatibility between the query and individual keys:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

This formulation implicitly assumes that all keys provide independent information. It is blind to the **intrinsic geometry of the key set itself**. Specifically, it fails to account for **redundancy**.

In high-dimensional semantic spaces, keys often cluster into dense groups—repeated tokens, near-synonyms, or recurring functional patterns. We call these **grouped keys**.

The Mechanism of Hijacking

These grouped keys exploit a vulnerability in the softmax normalization known as **probability hijacking**.

1. **Being Grouped**: Systematic redundancy means many keys $\{k_1, \ldots, k_m\}$ occupy the same region in vector space. Consequently, they all produce the same or similar dot product with the query: $q \cdot k_i \approx \text{constant}$.
2. **The Hijack**: The softmax function sums the exponentials of these scores in its denominator. A cluster of 50 mediocre keys—simply by virtue of being numerous—can accumulate a massive aggregate probability mass, swamping a single, highly relevant "unique" key.

For example, consider a single critical key k_{unique} with a high relevance score $e^{q \cdot k_u} = 20$, and a cluster of 50 redundant keys with low relevance scores $e^{q \cdot k_i} = 1$. The attention weights become:

$$\text{Weight}(k_{\text{unique}}) = \frac{20}{20 + \underbrace{(1 + \cdots + 1)}_{\text{50 times}}} = \frac{20}{70} \approx \mathbf{0.28}$$

$$\text{Weight}(\text{Cluster}) = \frac{50}{70} \approx \mathbf{0.71}$$

Despite the unique key being **20× more relevant** than any individual cluster member, the redundant group dominates the attention mechanism by sheer volume.

The model is forced to over-attend to the *frequency* of a signal rather than its *information content*. It wastes capacity retrieving the same redundant feature 50 times, while drowning out the singular, critical signal that appears only once.

The Mathematical Solution: Magnitude

To fix this, we need a way to measure the **"Effective Number of Points"** in a set.

Intuitively, if you have 3 key vectors:

- If they are all far apart (orthogonal), they provide **3 units of information**.
- If they are all identical (clones), they provide only **1 unit of information**.
- If they are somewhat similar, they provide somewhere between **1 and 3 units**.

The mathematical tool that captures this continuous "effective count" is called **Magnitude**. It assigns a weight μ_j to each key representing its unique contribution.

How It Works: The Weighting Equation

The core mechanism is a linear system that solves for the "uniqueness weight" of each key.

$$Z\boldsymbol{\mu} = \mathbf{1}$$

Here is exactly what each component represents:

1. Z **(The Similarity Matrix)**: This is an $N \times N$ matrix where entry Z_{ij} is a number between 0 and 1. It measures the similarity between key i and key j.
   - $Z_{ij} = 1$ means the keys are identical.
   - $Z_{ij} \approx 0$ means the keys are completely different (orthogonal).
2. $\boldsymbol{\mu}$ **(The Unknown Weights)**: This is a single column vector containing one weight for **every** key in the sequence: $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_N]^\top$. In the summation formula, we use the index i to denote the "current key" whose constraint we are checking, and the index j to iterate over all its neighbors.

3. **1 (The Unit Constraint)**: This is a vector of all ones. It acts as the target for every row.

Visualizing the Matrix System

To see the relationship between i and j, let's look at the full system for 3 keys ($N = 3$):

$$\begin{bmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{21} & Z_{22} & Z_{23} \\ Z_{31} & Z_{32} & Z_{33} \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

- **Row 1 (i=1)**: $Z_{11}\mu_1 + Z_{12}\mu_2 + Z_{13}\mu_3 = 1$. The first key ($i = 1$) must balance its own weight μ_1 with the weighted similarity of neighbors 2 and 3.
- **Row 2 (i=2)**: $Z_{21}\mu_1 + Z_{22}\mu_2 + Z_{23}\mu_3 = 1$. The second key ($i = 2$) has its own constraint, summing over all neighbors $j \in \{1, 2, 3\}$.

What the Equation Does:

The equation imposes a strict constraint on every single key i in the set:

$$\underbrace{\mu_i \cdot Z_{ii}}_{\text{Self-Contribution}} + \underbrace{\sum_{j \neq i} \mu_j \cdot Z_{ij}}_{\text{Contribution from Neighbors}} = 1$$

In plain English: "The sum of my own weight plus the weighted similarity of all my neighbors must exactly equal 1."

This forces a trade-off:

- **If a key has no neighbors ($Z_{ij} \approx 0$):** (Remember, Z_{ij} is the similarity calculated from distance, typically $e^{-\text{distance}^2}$, so being far apart means $Z \approx 0$). The second term disappears. The equation simplifies to $\mu_i \cdot 1 = 1$, so μ_i must be **1**. The key retains full weight.
- **If a key has many identical neighbors ($Z_{ij} \approx 1$):** The second term becomes very large because there are many neighbors contributing to the sum. To keep the total equal to 1, the weights μ must essentially strictly decrease. Specifically, if there are N identical keys, their weights must drop to $1/N$ so that their sum stays equal to 1.

By solving this system, we mathematically deduce exactly how redundant each key is relative to the entire group. The sum of these weights, $|X| = \sum \mu_i$, is the **Magnitude** of the set. It tells us truly how much information is present.

The Choice of Similarity Kernel

To make this system work in practice, we must choose a specific formula for the similarity Z_{ij}. We cannot just use any function; we need one that guarantees the linear system $Z\boldsymbol{\mu} = \mathbf{1}$ is actually solvable.

We use the **Gaussian Kernel**:

$$Z_{ij} = e^{-t \cdot \|x_i - x_j\|^2}$$

This specific choice is critical because the Gaussian kernel is **Strictly Positive Definite (SPD)**.

In simple terms, this property ensures that the similarity matrix Z is always **invertible**. Without it, if two keys were even slightly similar, the math could "break" (resulting in divide-by-zero errors or infinite solutions). The Gaussian kernel guarantees that as long as your tokens aren't perfect clones, there is always exactly one stable, unique answer for the weights $\boldsymbol{\mu}$.

(If you want to dive deeper into the technical mechanics, research: **"Why is the Gaussian (RBF) kernel strictly positive definite?"**, **"How does the SPD property guarantee matrix invertibility?"** and **"Numerical stability of solving linear systems with SPD matrices."**).

With this stable foundation, we can now build the full attention mechanism.

Key Properties

Property	**Statement**
Redundancy suppression	Points in dense clusters receive *lower* μ_j
Uniqueness amplification	Isolated/boundary points receive *higher* μ_j
Information Additivity	Total information is the sum of unrelated parts.
Diversity Limit	$\|X\|$ represents the absolute maximum variety.

The magnitude weight μ_j answers: **"How much unique geometric information does point j contribute, given all the other points?"**

Magnitude Attention: The Construction

Step 1 — Key-Space Geometry (Gaussian)

Given keys $\{k_1, \ldots, k_n\}$, we construct the similarity matrix Z. Unlike standard attention which uses a fixed dot product, we use a Gaussian kernel with a **learnable scale** t:

$$Z_{jl} = \exp\left(-t \cdot \frac{\|k_j - k_l\|^2}{d_k}\right)$$

This parameter t acts as a **geometrical resolution control**. By making t learnable, the model can autonomously tune the “sharpness” of its similarity metric:

- A **high** t creates a strict filter where only extremely close vectors are treated as redundant.
- A **low** t creates a broader filter, allowing the model to suppress “near-synonyms” or semantically related clusters that aren’t exact copies.

Essentially, the model learns the optimal radius at which two pieces of information should be considered “the same,” tuning the sensitivity of the suppression mechanism to the specific dataset.

Step 2 — Solving for Weights (Iterative)

To find $\boldsymbol{\mu}$, we solve the system $Z\boldsymbol{\mu} = \mathbf{1}$ using **Truncated Conjugate Gradient (CG)**.

Instead of a costly $O(N^3)$ matrix inversion, CG finds the optimal weights in just 3–5 iterations. This keeps the complexity at $O(N^2)$, matching the overhead of standard attention.

When keys are nearly identical, the system can become numerically unstable. We use **Tikhonov regularization** $(Z + \epsilon I)$ to ensure the solver always finds a unique, stable solution without gradients “exploding” during training.

Step 3 — Magnitude Gating

Once we have the uniqueness weights $\boldsymbol{\mu}$, we apply a **Magnitude Gate** to the values (V) before the attention sum.

$$\text{Gate}_j = \sigma(\beta \cdot \mu_j + \gamma)$$

$$\text{Output} = \text{Attn}(Q, K, (\text{Gate} \cdot V))$$

Unlike methods that just shift attention scores (which only change probabilities), this **physically dampens** the signal mass. By multiplying each value vector V_j by the gate before the sum, redundant tokens are scaled down toward zero. They effectively “shrink” or vanish from the hidden state, preventing their redundant features from accumulating and over-powering the unique information in the final output.

Consider a cluster of 50 duplicate tokens. Their magnitude weights will be $\mu_j = 1/50 = 0.02$. If the Magnitude Gate passes this weight through, the contribution of the entire cluster becomes:

$$\sum_{\text{50 tokens}} \text{Score} \cdot (0.02 \cdot V) = 50 \cdot (\text{Score} \cdot 0.02 \cdot V) = \mathbf{1.0} \cdot \text{Score} \cdot V$$

The sheer volume of the 50 tokens is mathematically collapsed, forcing the model to treat the entire cluster as just **1 unit of information**.

The learnable parameters (β, γ) allow the model to decide how aggressively to suppress redundancy. It learns to "trust" the geometric uniqueness weights only where they actually improve performance.