Vector Search

The Innovation Game Labs

December, 2025

1 Introduction

Vector search (also called approximate nearest neighbor search) is the task where, given 2 sets of vectors with, a database set and a query set, find for each query vector a nearby vector in the database set, such that the mean distance between the query vectors and their corresponding vector in the database is within a threshold value. Vector search is a fundamental problem in information retrieval and modern machine learning systems, underlying applications such as recommendation, semantic search, large-scale retrieval, and similarity matching in high-dimensional spaces.

2 TIG Challenge

The TIG challenge will run on GPUs. Let $\Omega \subset \mathbb{R}^{\text{dim}}$. The task is: given a set of query vectors $\mathcal{Q} = \{q_i\}_i \subset \Omega$ and a set of database vectors $\mathcal{D} = \{x_j\}_j \subset \Omega$, for each query $q_i \in \mathcal{Q}$ find a neighbor in the database set $x(q_i) \in \mathcal{D}$ to minimise the average distance

$$\frac{1}{|Q|} \sum_{i=1}^{|Q|} \|q_i - x(q_i)\|.$$

In TIG we set the dimension $\text{dim} = 250$, and set $\Omega = [-1, 1]^{250}$.

3 Random Instance Generation

All data instances are synthetically generated using a controlled *Gaussian mixture model*, designed to simulate realistic clustered structures while enabling fine-grained control over instance complexity. The size of the problem is determined by following this table of real-world datasets:

Dataset	Dimension	Database size	Queries
UQ-V	256	1,000,000	10,000
Msong	420	992,272	200
Audio	192	53,387	200
SIFT1M	128	1,000,000	10,000
GIST1M	960	1,000,000	1,000
Crawl	300	1,989,995	10,000
GloVe	100	1,183,514	10,000
Enron	1,369	94,987	200

Table 1: Summary of the eight datasets used in our experiments.

The dimension being 250 is a reasonable choice because techniques such as dimensionality reduction are typically employed to handle larger dimensions; by selecting a moderate dimension size, TIG eliminate the need for this step.

Let M be the number of database vectors and N the number of query vectors. The number of database vectors scales with the number of queries. The ratio is kept fixed at

$$\frac{M}{N} = 100,$$

which is in line with real-world data. In particular, TIG set

$$M = \lfloor 100 \cdot N \rfloor .$$

The number of query vectors N is a difficulty parameter controlled by the Benchmarkers. The domain which vectors are generated in is

$$[-1, 1]^{\text{dim}},$$

this allows vector directions to vary. Since real-world data is typically *clustered* TIG employs Gaussian clusters to generate the data. The size of the clusters, is guided by several frequently benchmarked datasets:

Dataset	Domain	# Points	# Classes	Points/Cluster
MNIST	Handwritten digits	70 000	10	7 000
CIFAR-100	Tiny images (fine-grained)	60 000	100	600
SVHN	House-number digits (real)	99 289	10	9 928
ImageNet-1k	Natural images	1 281 167	1 000	∼1 281
Tiny ImageNet	Subset of ImageNet	100 000	200	500
VGGFace2	Face recognition	3 310 000	9 131	∼363
Google Speech Cmds v2	Audio classification	105 829	35	∼3 023
Wikipedia (small cats.)	Sentences by topic	∼30 000	13	∼2 308

Table 2: Properties of commonly used benchmark datasets.

From this TIG infers that a reasonable number of vectors to be in a cluster is $r_{\text{clust}} = 700$, that is, if C is the number of clusters, then

$$\frac{N}{C} \;\approx\; r_{\text{clust}} \;=\; 700.$$

To promote variation in challenge instances TIG introduces a small random perturbation to the cluster count

$$C \;=\; \left\lfloor (1+\delta)\,\frac{M}{r_{\text{clust}}} \right\rfloor, \qquad \delta \sim \text{Unif}[-0.05, 0.05],$$

Clusters will *not* contain the same number of points—real datasets are imbalanced. In MNIST, digits "1" and "0" appear more often than "5" or "9"; in Wikipedia articles, broad topics such as "sports" dominate niche ones like "theology". We now describe how TIG determines the size distribution of the cluster sizes. Let K_i denote the size of a cluster i. For every cluster i draw

$$K_i \;\sim\; \text{LogNormal}(\mu, \sigma^2),$$

with (μ, σ^2) chosen so that $\mathbb{E}[K_i]$ matches the desired points-per-cluster ratio. This is done in the following way, recall that a log-normal random variable $K \sim \text{LogNormal}(\mu, \sigma^2)$ satisfies

$$\mathbb{E}[K] \;=\; e^{\,\mu+\sigma^2/2}, \qquad \text{Var}[K] \;=\; \big(\mathbb{E}[K]\big)^2 \left(e^{\sigma^2} - 1\right).$$

Hence the standard deviation of K is

$$\sqrt{\text{Var}[K]} \;=\; \mathbb{E}[K]\,\sqrt{e^{\sigma^2} - 1}.$$

The σ^2 of the lognormal is set to $\sigma^2 = 0.2$, so the standard deviation scales with the mean as

$$\mathbb{E}[K]\,\sqrt{e^{0.2} - 1} \;\approx\; \mathbb{E}[K] \times 0.47.$$

Choosing

$$\mu \;=\; \log\bigl(r_{\text{clust}}\bigr) - \frac{\sigma^2}{2}$$

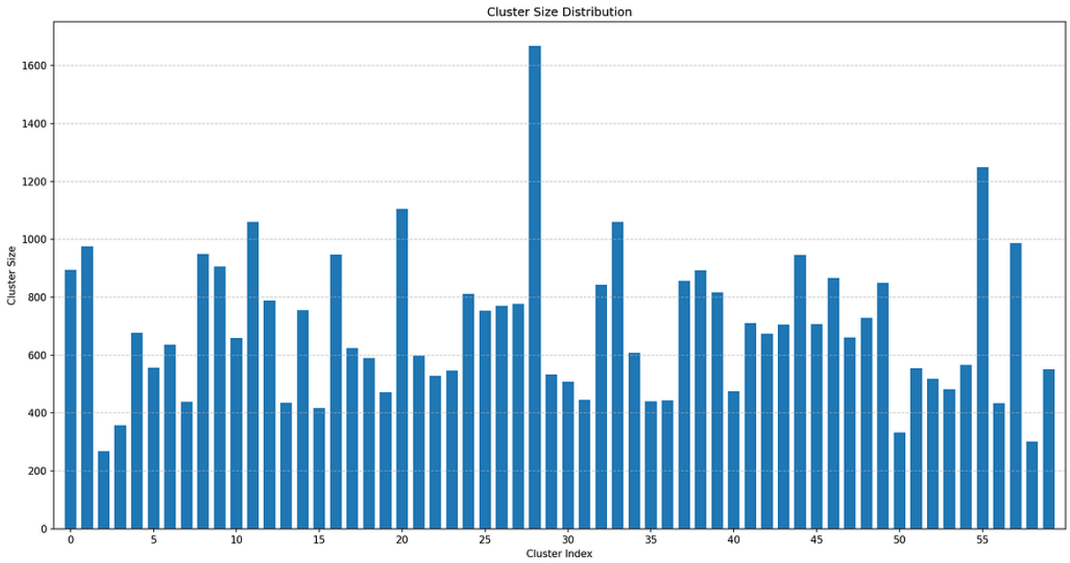
ensures that

$$\mathbb{E}[K] \;=\; r_{\text{clust}},$$

as desired. When sampling a database or query point, the instance generations draws from cluster i with probability

$$\frac{K_i}{\sum_{j=1}^{\#\text{Clusters}} K_j}.$$

This distribution is skewed, resulting in many small clusters, some medium-sized clusters, and only a few large clusters. The following is an example of 60 cluster sizes when sampled in this manner



Next we detail how TIG choose the location and shape of the clusters. Cluster vectors are sampled from a *truncated* multivariate Gaussian

$$\mathcal{N}_{[-1,1]^d}\bigl(\bar{\mu}, \Sigma\bigr),$$

whose support is restricted to the hyper-hypercube $[-1,1]^d$. Each cluster mean $\bar{\mu}$ is drawn uniformly from $[-1,1]^d$. Because real-world data is anisotropic, the protocol takes

$$\Sigma \;=\; \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_d^2 \end{pmatrix},$$

allowing the eigenvalues σ_i^2 to vary. Sampling is cheap: to draw $\boldsymbol{x} \sim \mathcal{N}_{[-1,1]^d}(\bar{\mu}, \Sigma)$ one just has to sample each coordinate independently,

$$x_j \;\sim\; \mathcal{N}_{[-1,1]}\bigl(\bar{\mu}_j, \sigma_j^2\bigr).$$

Choosing the variances. To maintain diversity in cluster shapes TIG allows the variances to vary. For every cluster $k \in \{1, \ldots, \#\text{Clusters}\}$ first sample a *mean deviation* $\sigma(k)$ and a *range* $\varepsilon(k)$, then set

$$\sigma_i(k) \;\sim\; \text{Unif}\bigl[\sigma(k) - \varepsilon(k), \sigma(k) + \varepsilon(k)\bigr], \quad i = 1, \ldots, d.$$

Where the mean deviation and range are sampled for each cluster as follows,

$$\sigma(k) \;\sim\; \text{Unif}[1,\, 1.1], \qquad \varepsilon(k) \;\sim\; \text{Unif}[0,\, 0.05].$$

With these parameters the expected cluster overlap is roughly 8% (i.e. about 8% of points lie closer to a centroid other than their own).

4 Challenge Tracks.

Currently each of the challenge tracks defines a different number of query vectors which are fed into the challenge generation. That is for each track an instance of vector search is tuned through the parameter:

Query size N: Larger N increases computation cost.

Other potential track parameters which are not employed in TIG are :

- **Dimensionality dim**: Higher dim increases geometric complexity and exacerbates the curse of dimensionality.
- **Cluster complexity**: Controlled by both the number of clusters C (derived from average cluster size r_{clust}) and their variance spread $[\sigma_{\min}, \sigma_{\max}]$, as well as the anisotropy factor $\epsilon_{\max}$.
- **Query distribution**: Whether queries are sampled *in-distribution* (from database clusters) or *out-of-distribution* (from novel clusters) directly impacts the generalization challenge.

These could be used for other tracks. Moreover we could also define different distance functions to use in different tracks.

5 Asymmetry.

The problem exhibits strong asymmetry between solving and verifying:

- **Solving:** requires navigating high-dimensional geometry and clustered structure to efficiently return a set of points $\{x(q_j)\}_j$ such that each q_j $j = 1, \dots N$ we have that $x(q_j)$ approximately satisfies $\|x(q_j) - q_j\|$ is small. This may involve complex indexing structures, dimensionality reduction, or learning-based heuristics, all of which incur substantial runtime cost.
- **Verifying:** once the candidate result set $\{x(q_j)\}_j$ is submitted, the verifier can simply compute the Euclidean distances between each point $x(q_j)$ and the corresponding query q_j, and check the mean distance $\frac{1}{N}\sum_{j=1}^{N} \|x(q_j) - q_j\|$ holds. Since the verifier has full access to the instance and the query, this check takes negligible time.