Welcome to Faiss Documentation

Faiss

Faiss is a library for efficient similarity search and clustering of dense vectors. It contains algorithms that search in sets of vectors of any size, up to ones that possibly do not fit in RAM. It also contains supporting code for evaluation and parameter tuning.

Faiss is written in C++ with complete wrappers for Python. Some of the most useful algorithms are implemented on the GPU. It is developed primarily at FAIR, the fundamental AI research team of Meta.

What is similarity search?

Given a set of vectors x_i in dimension d, Faiss builds a data structure in RAM from it. After the structure is constructed, when given a new vector **x** in dimension **d** it performs efficiently the operation:

$$j = argmin_i \|x - x_i\|$$

where is the Euclidean distance ().

In Faiss terms, the data structure is an *index*, an object that has an *add* method to add vectors. Note that the ’s are assumed to be fixed.

Computing the argmin is the *search* operation on the index.

This is all what Faiss is about. It can also:

- return not just the nearest neighbor, but also the 2nd nearest, 3rd, …, k-th nearest neighbor
- search several vectors at a time rather than one (batch processing). For many index types, this is faster than searching one vector after another
- trade precision for speed, ie. give an incorrect result 10% of the time with a method that’s 10x faster or uses 10x less memory

- perform maximum inner product search instead of minimum Euclidean search. There is also limited support for other distances (L1, Linf, etc.).
- return all elements that are within a given radius of the query point (range search)
- store the index on disk rather than in RAM.
- index binary vectors rather than floating-point vectors
- ignore a subset of index vectors according to a predicate on the vector ids.

Install

The recommended way to install Faiss is through Conda:

```
$ conda install -c pytorch faiss-cpu
```

The `faiss-gpu` package provides CUDA-enabled indices:

```
$ conda install -c pytorch faiss-gpu
```

Note that either package should be installed, but not both, as the latter is a superset of the former.

Research foundations of Faiss

Faiss is based on years of research. Most notably it implements:

- The inverted file from "Video google: A text retrieval approach to object matching in videos.", Sivic & Zisserman, ICCV 2003. This is the key to non-exhaustive search in large datasets. Otherwise all searches would need to scan all elements in the index, which is prohibitive even if the operation to apply for each element is fast
- The product quantization (PQ) method from "Product quantization for nearest neighbor search", Jégou & al., PAMI 2011. This can be seen as a lossy compression technique for high-dimensional vectors, that allows relatively accurate reconstructions and distance computations in the compressed domain.
- The three-level quantization (IVFADC-R aka *IndexIVFPQR*) method from "Searching in one billion vectors: re-rank with source coding", Tavenard & al., ICASSP'11.
- The inverted multi-index from "The inverted multi-index", Babenko & Lempitsky, CVPR 2012. This method greatly improves the speed of inverted indexing for fast/less accurate operating points.

- The optimized PQ from "Optimized product quantization", He & al, CVPR 2013. This method can be seen as a linear transformation of the vector space to make it more amenable for indexing with a product quantizer.
- The pre-filtering of product quantizer distances from "Polysemous codes", Douze & al., ECCV 2016. This technique performs a binary filtering stage before computing PQ distances.
- The GPU implementation and fast k-selection is described in "Billion-scale similarity search with GPUs", Johnson & al, ArXiv 1702.08734, 2017
- The HNSW indexing method from "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs", Malkov & al., ArXiv 1603.09320, 2016
- The in-register vector comparisons from "Quicker ADC : Unlocking the Hidden Potential of Product Quantization with SIMD", André et al, PAMI'19, also used in ["Accelerating Large-Scale Inference with Anisotropic Vector Quantization" <https://arxiv.org/abs/1908.10396>`_, Guo, Sun et al, ICML'20.
- The binary multi-index hashing method from "Fast Search in Hamming Space with Multi-Index Hashing", Norouzi et al, CVPR'12.
- The graph-based indexing method NSG from *"Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph"* <https://arxiv.org/abs/1707.00143>`_, Cong Fu et al, VLDB 2019.
- The Local search quantization method from "Revisiting additive quantization", Julieta Martinez, et al. ECCV 2016 and "LSQ++: Lower running time and higher recall in multi-codebook quantization", Julieta Martinez, et al. ECCV 2018.
- The residual quantizer implementation from "Improved Residual Vector Quantization for High-dimensional Approximate Nearest Neighbor Search", Shicong Liu et al, AAAI'15.

A general paper about product quantization and related methods: "A Survey of Product Quantization", Yusuke Matsui, Yusuke Uchida, Hervé Jégou, Shin'ichi Satoh, ITE transactions on MTA, 2018.

The overview image below is from that paper (click on the image to enlarge it):

Image credit: Yusuke Matsui, thanks for allowing us to use it!

Methods that are implemented in Faiss are highlighted in red.

Key to all references

Legal

See the Terms of Use and Privacy Policy.