



IJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: <https://doi.org/10.22214/ijraset.2026.77975>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

Cluster: A Lightweight Rust-Accelerated Clustering Toolkit

Soumyadip Sarkar

Abstract: We study *Cluster*, a compact clustering toolkit implemented in Rust and exposed to Python via a thin extension module. *Cluster* targets a pragmatic design point: implement classical clustering algorithms with a Python-first surface that accepts and returns NumPy arrays, while remaining minimal-dependency and performance-conscious. We present a formal specification of the *Cluster* API contracts, map each implemented algorithm to primary literature, and provide proof sketches for key theoretical properties (objective descent, termination, statistical consistency where applicable, and EM monotonicity). We also propose a rigorous experimental evaluation plan spanning synthetic and real benchmarks, and compare *Cluster*'s scope and trade-offs to mainstream clustering libraries. The code can be found at <https://github.com/alphavelocity/cluster>

I. INTRODUCTION

Cluster is a Rust-accelerated clustering toolkit exposed to Python via PyO3/maturin [1, 4, 5]. Its core contribution is not a new clustering objective, but an engineering and packaging point: provide classical unsupervised learning primitives with low overhead and predictable memory layout. This paper delivers: (i) an API and mathematical contract for each operator, (ii) theory-oriented documentation (definitions, proof sketches, complexity), (iii) pseudocode aligned to the implementations, (iv) an evaluation plan and dataset recommendations, (v) limitations and open problems, and (vi) a code-heavy appendix with runnable examples and derivations.

II. BACKGROUND AND MOTIVATION

Clustering remains a foundational unsupervised primitive for exploratory analysis, representation learning pipelines, and as a preprocessing step for downstream models. In practice, users frequently want (a) a stable API, (b) predictable performance, (c) low dependency surface, and (d) sufficient algorithmic breadth to cover common regimes (spherical/Euclidean partitions, streaming, density structure, hierarchical summaries, and mixture models). *Cluster* aims at this “classical essentials” layer [1], using Rust for compute kernels and PyO3/maturin for Python integration [3, 5].

We emphasize a critical methodological point: *algorithmic convergence* (e.g., Lloyd termination) differs from *statistical consistency* (population recovery as $\square \rightarrow \infty$). Both are discussed where relevant [18, 20].

III. PROBLEM SETTING AND API CONTRACTS

A. Data model and notation

Let $\square \in \mathbb{R}^{\square \times \square}$ be a dataset with rows $\square_{\square} \in \mathbb{R}^{\square}$. *Cluster*'s Python layer (per repo inspection) coerces inputs to float64 contiguous arrays, hence we assume \square is stored as a contiguous buffer in row-major order.

A *hard clustering* is a label map $\ell: \{1, \dots, \square\} \rightarrow \{0, \dots, \square - 1\}$. A *noise-aware clustering* extends the codomain to $\{-1\} \cup \{0, \dots, \square - 1\}$ where -1 denotes noise (DBSCAN/OPTICS conventions) [13, 7]. A *soft clustering* is a responsibility matrix $\square \in [0, 1]^{\square \times \square}$ with $\sum_{\square=1}^{\square} \square_{\square \square} = 1$.

B. Distance functions and preprocessing

Cluster supports at least the following metrics (per repo inspection and documented API):

$$\begin{aligned} \square_2(\square, \square) &= \|\square - \square\|_2, \\ \square_{\cos}(\square, \square) &= 1 - \frac{\langle \square, \square \rangle}{\|\square\|_2 \|\square\|_2}, \end{aligned}$$

with standard conventions for zero vectors and optional normalization for cosine workflows. Cosine-based pipelines often operate on the unit sphere; therefore optional row normalization $\square \leftarrow \square / \|\square\|_2$ is treated as part of the operator specification.

C. *Output schemas and invariants*

Cluster methods return NumPy arrays and/or dictionaries with keys such as: labels, centers, inertia, reachability, ordering, weights, means, covars, resp, depending on the algorithm family [1].

IV. ALGORITHMS IMPLEMENTED IN CLUSTER

A. *K Means and K Means++ initialization*

The KMeans objective is the within-cluster sum of squares:

$$J(\mu, \mu) = \sum_{i=1}^n \mu_i \min_{c \in \{1, \dots, K\}} \|\mu_i - \mu_c\|_2^2,$$

with optional weights $\mu_i \geq 0$. Cluster uses KMeans++ style seeding [8] and Lloyd-style alternating updates [18].

B. *MiniBatchKMeans*

Mini-batch KMeans updates centers using small batches, as in web-scale settings [22]. Cluster also supports a streaming partial_fit-style interface (repo/API inspection). This is best interpreted as a stochastic approximation to the batch objective.

C. *BisectingKMeans*

Bisecting KMeans builds K clusters via repeated 2-way splits, typically selecting a cluster to split by a variance/SSE criterion (repo inspection). This yields a top-down hierarchical decomposition and can be advantageous when K is moderate and interpretability matters.

D. *DBSCAN and OPTICS*

DBSCAN defines clusters via density connectivity under parameters $(\epsilon, \text{minPts})$ [13]. OPTICS generalizes across scales by producing a reachability ordering encoding DBSCAN-like cluster structures for varying ϵ [7].

E. *Affinity Propagation*

Affinity Propagation performs exemplar-based clustering by message passing on pairwise similarities [14]. Damping is used to stabilize iterations in practice.

F. *BIRCH*

BIRCH summarizes streaming data via a CF-tree (cluster feature tree) and can optionally refine subclusters via a final clustering stage [26, 27].

G. *Diagonal-covariance Gaussian Mixture Models*

Cluster implements a diagonal-covariance Gaussian mixture model (GMM) fitted by EM [11]. Let μ_i be mixture weights, $\mu_i \in \mathbb{R}^D$ means, and $\Sigma_i = \text{diag}(\sigma_{i1}^2, \dots, \sigma_{iD}^2)$. The log-likelihood is

$$\mathcal{L}(\mu) = \sum_{i=1}^n \mu_i \log \left(\sum_{c=1}^K \mu_c \mu(\mu_i | \mu_c, \Sigma_c) \right).$$

H. *Agglomerative hierarchical clustering and Ward linkage*

Hierarchical agglomerative clustering (HAC) produces a dendrogram via repeated merges. Ward’s method merges the pair that minimally increases within-cluster variance (Euclidean-only) [24].

I. *Internal validation metrics*

Cluster includes internal indices: Silhouette [21], Calinski–Harabasz [9], and Davies–Bouldin [10]. A survey context is given in [15].

J. Algorithmic complexity summary

Table 1 summarizes asymptotic cost under Cluster’s exact distance scans.

Asymptotic time and space complexity (dominant terms) for Cluster kernels.

Method	Time (approx.)	Space (approx.)
KMeans (per iter)	$O(nkd)$	$O(nd + kd + n)$
MiniBatchKMeans (per step)	$O(bkd)$	$O(kd + k)$
BisectingKMeans	$\approx O(\sum 2\text{-KMeans splits})$	$O(nd + kd)$
DBSCAN	$O(n^2d)$	$O(n)$ (+ neighbor lists)
OPTICS	$O(n^2d + n \log n)$	$O(n)$ (+ heap)
Affinity Propagation	$O(n^2d + Tn^2)$	$O(n^2)$
BIRCH	$O(n \cdot \text{depth} \cdot d)$	$O(B \cdot d)$ (tree)
GMM (per EM iter)	$O(nkd)$	$O(nk + kd)$
HAC	$O(n^3)$	$O(n^2)$
Silhouette	$O(m^2d)$ for m non-noise	$O(mk)$ accumulators

V. THEORETICAL PROPERTIES

A. KMeans: objective descent and finite termination

1) Claim (monotone descent).

Under Lloyd updates with fixed k and weights $w_i \geq 0$, alternating (i) assignment to nearest centers and (ii) recomputation of each μ_c as the (weighted) mean of its assigned points does not increase Φ .

2) Sketch.

Given centers, assignment minimizes Φ pointwise. Given assignments, the weighted mean minimizes squared error in each cluster (Appendix derivation). Thus each step is non-increasing [18].

3) Claim (finite termination).

If ties are resolved deterministically, Lloyd iterations terminate in finitely many steps at a partition that is locally optimal under single-point reassignment.

4) Sketch.

There are finitely many partitions of n points into k labels; the objective decreases whenever the partition changes, hence termination occurs in finite steps.

B. KMeans++ approximation and statistical consistency

KMeans++ seeding is an $O(\log k)$ -approximation in expectation to the optimal k -means objective [8]. For statistical consistency of global empirical k -means minimizers under standard conditions, see Pollard [20].

C. MiniBatchKMeans: stochastic optimization interpretation

Mini-batch updates can be interpreted as stochastic optimization of Φ with variance reduction; the original web-scale motivation and empirical behavior are described in [22]. Convergence guarantees depend on step-size schedules and regularity; we give practical guidance in the appendix.

D. EM for diagonal GMMs: monotonicity and stationary convergence

1) Claim (likelihood ascent).

Each EM iteration is guaranteed to not decrease the observed-data likelihood.

2) Sketch.

EM constructs a lower bound via Jensen’s inequality on the complete-data log-likelihood, then maximizes it in the M-step; see [11]. Under mild regularity conditions, EM converges to stationary points [25].

VI. ALGORITHM PSEUDOCODE

A. KMeans with KMeans++ seeding

Algorithm 1 KMeans with KMeans++ seeding and Lloyd updates

Require: Dataset $X \in \mathbb{R}^{n \times d}$, k , n_init , max_iter , tolerance τ

Ensure: Centers $\mu \in \mathbb{R}^{k \times d}$, labels $\ell \in \{0, \dots, k-1\}^n$

```

1: for  $r = 1$  to  $n\_init$  do
2:   Initialize  $\mu^{(0)} \leftarrow$  KMeans++( $X, k$ ) [8]
3:   for  $t = 0$  to  $max\_iter-1$  do
4:      $\ell^{(t+1)}(i) \leftarrow \arg \min_c \|x_i - \mu_c^{(t)}\|_2^2$ 
5:      $\mu_c^{(t+1)} \leftarrow$  mean of  $\{x_i : \ell^{(t+1)}(i) = c\}$  (weighted if  $w_i$  present)
6:     if  $\max_c \|\mu_c^{(t+1)} - \mu_c^{(t)}\|_2 \leq \tau$  then
7:       break
8:     end if
9:   end for
10: end for
11: Return best run by minimal inertia

```

B. DBSCAN

Algorithm 2 DBSCAN (density-based clustering)

Require: Dataset X , radius ε , $minPts$

Ensure: Labels $\ell \in \{-1, 0, \dots\}^n$

```

1: Mark all points unvisited; set cluster_id  $\leftarrow 0$ 
2: for  $i = 1$  to  $n$  do
3:   if  $i$  visited then
4:     continue
5:   end if
6:    $N_\varepsilon(i) \leftarrow \{j : d(x_i, x_j) \leq \varepsilon\}$ 
7:   if  $|N_\varepsilon(i)| < minPts$  then
8:      $\ell(i) \leftarrow -1$  ▷ noise
9:   else
10:    Expand cluster_id by BFS/queue over density-reachable points [13]
11:    cluster_id  $\leftarrow$  cluster_id + 1
12:   end if
13: end for

```

C. EM for diagonal GMM

Algorithm 3 EM for diagonal-covariance Gaussian mixture

Require: Dataset X , components k , init (π, μ, Σ)

```

1: repeat
2:   E-step:  $R_{ic} \propto \pi_c \mathcal{N}(x_i | \mu_c, \Sigma_c)$ ; normalize over  $c$ 
3:   M-step:  $\pi_c \leftarrow \frac{1}{\sum_i w_i} \sum_i w_i R_{ic}$ 
4:    $\mu_c \leftarrow \frac{1}{\sum_i w_i R_{ic}} \sum_i w_i R_{ic} x_i$ 
5:    $\Sigma_c \leftarrow \text{diag} \left( \frac{1}{\sum_i w_i R_{ic}} \sum_i w_i R_{ic} (x_i - \mu_c)^2 \right) + \lambda I$ 
6: until lower bound improvement  $\leq tol$ 

```

VII. EXPERIMENTAL EVALUATION PLAN

We recommend an evaluation matrix covering (i) correctness on small datasets, (ii) calibrated comparisons to reference libraries, and (iii) scalability stress tests.

1) Datasets.

Table 2 lists a representative suite spanning classical tabular data and benchmark vision embeddings.

Suggested datasets and why they matter.

Dataset	Regime	Why
Iris (UCI) [6]	small, low-d	correctness, sanity checks
MNIST [17]	large, image	embedding clustering stress
Fashion-MNIST [2]	large, image	harder MNIST-like benchmark
Synthetic varying density	2D/HD	DBSCAN/OPTICS failure modes

2) Metrics.

When labels exist, report ARI/NMI [16, 23]. Always report internal indices (silhouette/CH/DB) with caveats [21, 9, 10, 15].

3) Baselines.

Compare against scikit-learn implementations and configuration-matched objectives [19]. For acceleration studies, include distance-bounding k-means variants such as Elkan’s method [12].

VIII. RELATED WORK AND POSITIONING

Cluster overlaps substantially with general-purpose clustering interfaces such as scikit-learn [19], but differs in its Rust-accelerated kernel strategy and minimal-dependency focus. A detailed feature comparison table is provided in Appendix.

IX. LIMITATIONS AND OPEN PROBLEMS

Cluster emphasizes minimal dependencies, but this can imply quadratic scalability for neighborhood-heavy routines without spatial indexing. Open problems include: (i) optional ANN backends for DBSCAN/OPTICS and validation metrics, (ii) additional covariance structures for GMMs, (iii) strict semantic alignment and compatibility layers with reference APIs, and (iv) multi-threaded kernels (where appropriate) under optional features (e.g., Rayon).

X. CONCLUSION

Cluster provides a compact Rust-based toolkit for classical clustering workflows in Python [1]. By mapping each operator to primary literature and stating explicit mathematical and API contracts, we aim to make the library easier to audit, benchmark, and extend.

REFERENCES

- [1] cluster (python package) — project description and feature list. <https://pypi.org/project/cluster/>, accessed 2026-03-01
- [2] Fashion-mnist: A mnist-like fashion product database. https://research.zalando.com/project/fashion_mnist/fashion_mnist/, accessed 2026-03-01
- [3] maturin (pyo3/mixed rust–python packaging tool) — repository and documentation. <https://github.com/PyO3/maturin>, accessed 2026-03-01
- [4] maturin user guide: Bindings. <https://www.maturin.rs/bindings>, accessed 2026-03-01
- [5] Pyo3: Rust bindings for the python interpreter. <https://pyo3.rs/main/>, accessed 2026-03-01
- [6] Iris dataset — uci machine learning repository. <https://archive.ics.uci.edu/dataset/53/iris> (1936). <https://doi.org/10.24432/C56C76>, donated 1988; DOI 10.24432/C56C76; Accessed 2026-03-01
- [7] Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD ’99). pp. 49–60. ACM (1999). <https://doi.org/10.1145/304181.304187>
- [8] Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA ’07). pp. 1027–1035. SIAM (2007), <https://research.google/pubs/k-means-the-advantages-of-careful-seeding/>
- [9] Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics 3(1), 1–27 (1974). <https://doi.org/10.1080/03610927408827101>
- [10] Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1(2), 224–227 (1979). <https://doi.org/10.1109/TPAMI.1979.4766909>
- [11] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B (Methodological) 39(1), 1–22 (1977). <https://doi.org/10.1111/j.2517-6161.1977.tb01600.x>
- [12] Elkan, C.: Using the triangle inequality to accelerate k-means. In: Proceedings of the 20th International Conference on Machine Learning (ICML 2003) (2003), <https://vvvvv.aaai.org/Library/ICML/2003/icml03-022.php>
- [13] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD’96). pp. 226–231 (1996), <https://proceedings.aaai.org/Library/KDD/1996/kdd96-037.php>



- [14] Frey, B.J., Dueck, D.: Clustering by passing messages between data points. *Science* 315(5814), 972–976 (2007). <https://doi.org/10.1126/science.1136800>
- [15] Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. *Journal of Intelligent Information Systems* 17(2–3), 107–145 (2001). <https://doi.org/10.1023/A:1012801612483>
- [16] Hubert, L., Arabie, P.: Comparing partitions. *Journal of Classification* 2(1), 193–218 (1985). <https://doi.org/10.1007/BF01908075>
- [17] LeCun, Y., Cortes, C., Burges, C.J.C.: The mnist database of handwritten digits. <https://yann.lecun.org/exdb/mnist/index.html>, accessed 2026-03-01
- [18] Lloyd, S.: Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28(2), 129–137 (1982). <https://doi.org/10.1109/TIT.1982.1056489>
- [19] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12(85), 2825–2830 (2011), <https://www.jmlr.org/papers/v12/pedregosa11a.html>
- [20] Pollard, D.: Strong consistency of k -means clustering. *The Annals of Statistics* 9(1), 135–140 (1981). <https://doi.org/10.1214/aos/1176345339>
- [21] Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20, 53–65 (1987). [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7)
- [22] Sculley, D.: Web-scale k -means clustering. In: *Proceedings of the 19th International Conference on World Wide Web (WWW 2010)*. pp. 1177–1178. ACM (2010). <https://doi.org/10.1145/1772690.1772862>
- [23] Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* 11(95), 2837–2854 (2010), <https://jmlr.org/beta/papers/v11/vinh10a.html>
- [24] Ward, J.H.: Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association* 58(301), 236–244 (1963). <https://doi.org/10.1080/01621459.1963.10500845>
- [25] Wu, C.F.J.: On the convergence properties of the EM algorithm. *The Annals of Statistics* 11(1), 95–103 (1983)
- [26] Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering method for very large databases. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD '96)*. pp. 103–114. ACM (1996). <https://doi.org/10.1145/233269.233324>
- [27] Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery* (1997), <https://research.ibm.com/publications/birch-a-new-data-clustering-algorithm-and-its-applications>



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)