Clustering Algorithm Complexity & Performance Guide

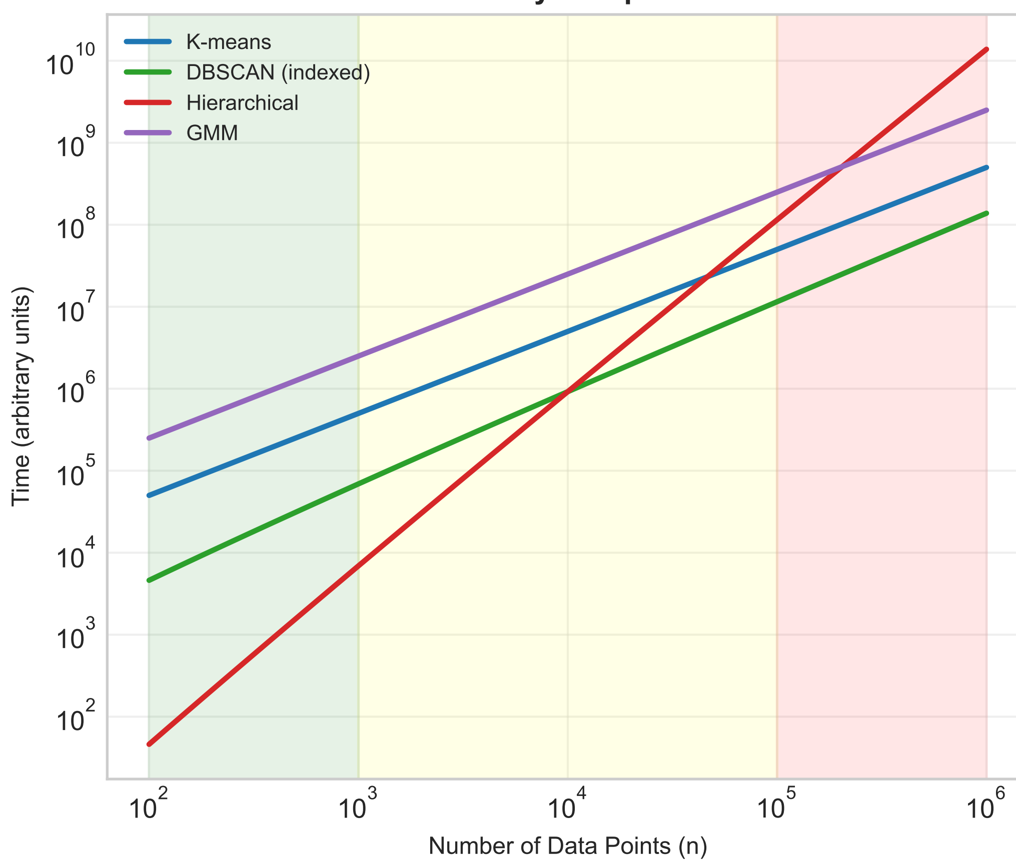
Algorithm Complexity Analysis

Big O Notation Comparison

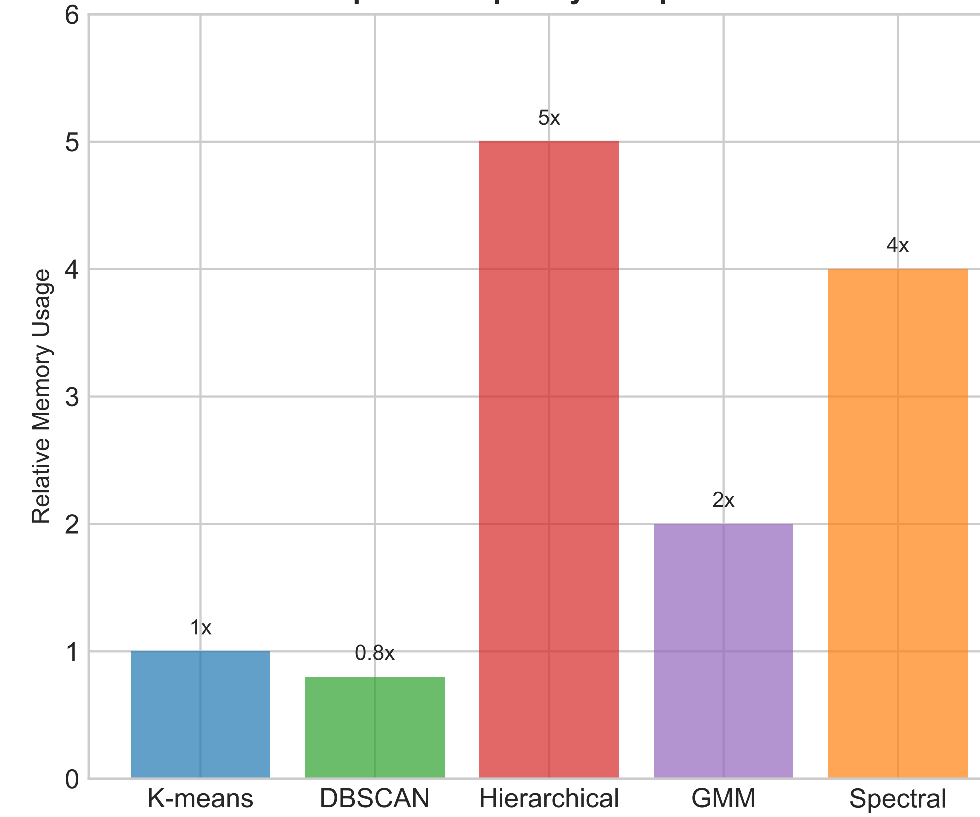
Algorithm	Time Complexity	Space Complexity	Scalability
K-means	O(n·k·i·d)	O(n·d + k·d)	Excellent
DBSCAN	O(n²) / O(n log n)*	O(n)	Good
Hierarchical	O(n³) / O(n² log n)*	O(n²)	Poor
GMM	O(n·k²·i·d)	O(k·d²)	Moderate
	O(n³)	O(n²)	Poor

Notation Guide:
n = number of data points
k = number of clusters
i = number of iterations
d = number of dimensions
* = with spatial index

Scalability Comparison



Space Complexity Comparison



Practical Recommendations

Small Data (<10K points)
→ Any algorithm works

Medium Data (10K-100K)
→ K-means or DBSCAN

Large Data (>100K)
→ MiniBatch K-means

High Dimensions (>50)
→ Consider PCA first

Real-time Requirements
→ Pre-computed K-means

Memory Constrained
→ Avoid Hierarchical

Optimization Techniques

MiniBatch K-means:
- Samples subset of data
- 10-100x faster on large data

Spatial Indexing (DBSCAN):
- KD-tree or Ball-tree
- O(n²) → O(n log n)

Dimensionality Reduction:
- PCA before clustering
- Reduces d in O(n·k·i·d)

Early Stopping:
- Monitor convergence
- Stop when stable

Implementation Complexity

Algorithm	Ease	Lines of Code*	Tuning
K-means	Easy	~50	Simple
DBSCAN	Moderate	~100	Tricky
Hierarchical	Easy	~30	Simple
GMM	Hard	~200	Complex
Spectral	Hard	~150	Complex

** Approximate implementation from scratch*

Performance Tips:
1. Profile before optimizing
2. Use vectorized operations
3. Consider approximate methods
4. Parallelize when possible

Choose algorithms based on data size, dimensionality, and performance requirements