Essay 2: Locality-Sensitive Hashing

Locality-Sensitive Hashing (LSH)  
--------------------------------  
  
Locality-Sensitive Hashing is a family of randomized algorithms for approximate nearest neighbor (ANN) search. Unlike JL, which preserves distances via linear maps, LSH preserves similarity probabilistically: similar points collide under hash functions with higher probability.  
  
\*\*Definition\*\*   
A family of hash functions H is (r1,r2,p1,p2)-sensitive if:   
- For any points u,v: if ||u-v|| ≤ r1 then P[h(u)=h(v)] ≥ p1.   
- If ||u-v|| ≥ r2 then P[h(u)=h(v)] ≤ p2.   
Here p1 > p2 ensures discrimination.  
  
\*\*Examples\*\*   
  
1. \*SimHash (for cosine similarity):\*   
 Pick random hyperplanes a ~ N(0,I). Define h(x) = sign(a·x).   
 For unit vectors u,v,   
 P[h(u)=h(v)] = 1 - θ/π, where θ = angle(u,v).   
 Thus Hamming distance between bit hashes estimates angular distance.  
  
2. \*p-stable LSH (for ℓ2):\*   
 Choose a ~ N(0,I), b ~ U(0,w).   
 h(x) = floor((a·x + b)/w).   
 For points u,v, probability of collision depends on ||u-v||₂. With multiple concatenated hashes (k per table, L tables), we amplify the gap between close and far pairs.  
  
\*\*Mathematics\*\*   
For SimHash:   
 P[h(u)=h(v)] = 1 - (arccos(⟨u,v⟩))/π.   
For p-stable LSH:   
 Difference a·(u-v) is distributed N(0,||u-v||²). Collision probability is:   
 P[h(u)=h(v)] = ∫ f\_{N(0,||u-v||²)}(t) \* max(0,1 - |t|/w) dt.  
  
\*\*Complexity\*\*   
With LSH, query time ~ O(n^ρ) where ρ < 1, instead of O(n). Storage ~ O(n^{1+ρ}). Parameters (k,L,w) control accuracy-speed tradeoff.  
  
\*\*Applications\*\*   
- Nearest neighbor search in high-D.   
- Duplicate detection, plagiarism detection.   
- Large-scale clustering and similarity joins.  
  
LSH highlights a different principle: we can abandon exact distance preservation and instead rely on collision probabilities to approximate geometry efficiently.