### Topics

+ review song similarity; print result in HTML

+ review movie similarity

+ kNN classification

+ kNN clustering

+ scalable/indexed search engine

# ------------------ clustering

### Clustering = classification without label

|  |
| --- |
| **package** tasks.july08\_knn\_clustering;  **public** **class** task1\_knn\_clustering  {  **public** **static** **void** main(String[] args)  {  Clustering<**double**[]> c = **new** MyClusteringEngine();  c.add(**new** **double**[] { 100, 0, 0 });  c.add(**new** **double**[] { 101, 0, 0 });  c.add(**new** **double**[] { 102, 0, 0 });  c.add(**new** **double**[] { 105, 0, 0 });    c.add(**new** **double**[] { 0, 301, 0 });  c.add(**new** **double**[] { 0, 309, 0 });  c.add(**new** **double**[] { 0, 304, 0 });  c.add(**new** **double**[] { 0, 305, 0 });  c.add(**new** **double**[] { 0, 307, 0 });    c.add(**new** **double**[] { 0, 0, 290 });  c.add(**new** **double**[] { 0, 0, 291 });  c.add(**new** **double**[] { 0, 0, 293 });  c.add(**new** **double**[] { 0, 0, 298 });      c.predictWithKmeans(3, 5);    System.***out***.println("=================");  **for**(**int** k=0; k<c.size(); k++)  {  System.***out***.println(c.getPoint(k) + " -> " + c.getLabel(k));  }  }  } |

### Function

|  |
| --- |
| **public** **void** predictWithKmeans(**int** kpar, **int** maxiter)  {  randomizeLabels(kpar);    **for**(**int** tt=0; tt<maxiter; tt++)  {  List<Cluster<T>> c = computeCenters(kpar);  relabelWithCenters(c);  }  }  **private** **void** relabelWithCenters(List<Cluster<T>> c)  {  **for**(**int** k=0; k<points.size(); k++)  {  **int** lk = nearestCenter(points.get(k), c);  labels.set(k, lk);  }  }  **private** **int** nearestCenter(T pk, List<Cluster<T>> c)  {  **double** dmin = Double.***MAX\_VALUE***;  **int** kmin = -1;    **for**(**int** k=0; k<c.size(); k++)  {  **double** dk = c.get(k).distance(pk);  **if**(dk < dmin) { dmin = dk; kmin = k; }  }    **return** kmin;  }  **private** **void** randomizeLabels(**int** kpar)  {  Random coin = **new** Random(197);    **for**(**int** k=0; k<points.size(); k++)  {  **int** lk = coin.nextInt(kpar);  labels.add(lk);  }  }  @SuppressWarnings("unchecked")  **private** List<Cluster<T>> computeCenters(**int** kpar)  {  List<Cluster<T>> res = **new** ArrayList<Cluster<T>>();  **for**(**int** k=0; k<kpar; k++) res.add((Cluster<T>) **new** ClusterDouble() );    **for**(**int** k=0; k<points.size(); k++)  {  **int** lk = labels.get(k);  T pk = points.get(k);  res.get(lk).add(pk);  }    **for**(**int** k=0; k<res.size(); k++)  res.get(k).average();      **return** res;  } |

### How to cluster movies / songs?

+ movies / songs -> structured data / categorical (no scaling, no adding)

+ vector -> numberical data (linear space, add, multiple)

# Scalabled/index search (lucene)

+ use index to search

|  |  |
| --- | --- |
| d1 | w1 w2 w3 |
| d2 | w3 w4 |
| d3 | w1 w2 |
| d4 | w4 w3 |
| d5 | w1 |

### The code for memory-based indexing

|  |
| --- |
| **package** tasks.july08\_scalabled\_indexed\_search;  **import** java.util.Map;  **import** java.util.Set;  **import** java.util.TreeMap;  **import** java.util.TreeSet;  **public** **class** test1\_indexed\_search {  **public** **static** **void** main(String[] args)  {  String[] docs = { "w1 w2 w3",  "w3 w4",  "w1 w2",  "w4 w3",  "w1",  };    Map<String, Set<String>> index = **new** TreeMap<String, Set<String>>();    **for**(String dk: docs)  {  System.***out***.println("====== " + dk.hashCode() + ": " + dk);  **for**(String wj: dk.split("\\s+"))  {  Set<String> sj = index.get(wj);  **if**(sj == **null**) index.put(wj, sj = **new** TreeSet<String>());  sj.add(dk.hashCode() + "");  }  }    **for**(String wj: index.keySet())  System.***out***.println(wj + " -> " + index.get(wj));    String q = "w1 w2";  Map<String, Integer> R = **new** TreeMap<String, Integer>();  **for**(String wj: q.split("\\s+"))  {  Set<String> sj = index.get(wj);  //System.out.println("Querying " + wj + " -> " + sj);    **if**(sj != **null**)  **for**(String dk: sj)  {  Integer ck = R.get(dk);  R.put(dk, ck==**null** ? 1 : ck+1);  //System.out.println(dk + ": " + R.get(dk));  }  }    **for**(String dk: R.keySet())  System.***out***.println(dk + ":" + R.get(dk));  }  } |