Recent Developments in Clustering

- Ben London (bal2123@columbia.edu)
- COMS W4772
- Prof. Tony Jebara
Abstract

- Part 1: Unsupervised
  - Implement / test \texttt{k-means++} algo
  - Extend \texttt{k-means++} technique to EM
    - Theoretical results?
    - Empirical results: improves EM
- Part 2: Semi-supervised
  - Implement / test \texttt{BoostCluster} algo
    - Empirical results: better than spectral?
Clustering

- Given set of N points in $\mathbb{R}^d$, partition into k clusters (groups/classes)
- Deterministic solution is in NP
- Many heuristics
- We have seen
  - Gradient descent: k-means, EM
  - Graph theory: spectral
- New!
  - Initialization (seeding)?
  - Boosting?
Initializing k-means

- Traditional approach: RANDOM
  - PRO: simple, efficient
  - CON: centroids sometimes overlap
  - Can we do better?

- Deterministic approach: Farthest-point heuristic
  - PRO: good for well-formed clusters
  - CON: sensitive to noise (outliers)

- Can we combine these two techniques?
**k-means++**

- Approximation method:
  - Heuristic algo
  - \(O(\log k)\)-competitive with optimal

- Minimize potential function:
  \[ \phi = \sum_{x \in X} \min_{c \in C} \| x - c \|^2 \]

- Algorithm:
  1) Initialize k clusters with \(D^2\) seeding
  2) Run **k-means**
D² Seeding

1) Select first centroid $c_1$ uniformly at random from $X$.
2) Calculate $D^2(x)$, for all $x$ in $X$. $D^2(x) = \|x - c_{closest}\|^2$
3) Select each successive centroid $c_i$ with probability

$$Pr[x \text{ chosen}] = \frac{D^2(x)}{\sum_{x \in X} D^2(x)}$$

4) Repeat steps 2 and 3 until all $k$ centroids have been selected
Initializing EM

- Can we apply $D^2$ seeding to EM?
- Empirical results:
  - Improves convergence time
  - Improves quality of converged solution (higher log-likelihood)
- Theoretical analysis is difficult
Semi-supervised

- Extremely relevant
- Partially labeled data
- Can be represented in the form of pairwise clustering constraints (NxN matrix)
 BoostCluster

- Semi-supervised clustering using boosting methodology
- Assumption: if a clustering satisfies the known pairwise constraints, then it is likely to satisfy the unknown pairwise constraints
- Uses iterative boosting technique to satisfy constraints
- Algorithm agnostic
  - Could use kNN, k-means, spectral, etc.
- Does not return classifier; only pairwise clusterings
Input
- $X$: $d \times n$ matrix for the input data
- $\mathcal{A}$: the given clustering algorithm
- $s$: the number of principal eigenvectors used for projection
- $S^+$: matrix for must-link pairs
- $S^-$: matrix for cannot-link pairs

Output: cluster memberships

Algorithm
- Initialize $K_{i,j} = 0$ for any $i, j = 1, 2, \ldots, n$.
- For $t = 1, 2, \ldots, T$
  - Compute $p_{i,j}$ and $q_{i,j}$ using (5) and (6).
  - Compute matrix $T$ using (10).
  - Compute the top $s$ eigenvectors and eigenvalues $\{(\lambda_i, v_i)\}_{i=1}^s$ of $T$.
  - Construct the projection matrix $P$ using (11), and generate the new data representation $X'$ by projecting the input data $X$ onto $P$.
  - Run the clustering algorithm $\mathcal{A}$ using the new data representation $X'$. Compute the matrix $\Delta$ with $\Delta_{i,j} = 1$ when $x_i$ and $x_j$ are grouped into the same cluster by $\mathcal{A}$, and zero otherwise.
  - Compute $\alpha$ using (13).
  - Update the kernel similarity matrix $K$ as $K + \alpha\Delta \rightarrow K$
- Run the clustering algorithm $\mathcal{A}$ with the kernel matrix $K$ (if $\mathcal{A}$ does not take a kernel similarity matrix as input, a data representation can be generated by the first $s + 1$ eigenvectors of the matrix $K$).
BoostCluster: High-level

- Loss function: 
  \[ L = \left( \sum_{i,j} S_{i,j}^+ \exp(-K_{i,j}) \right) \left( \sum_{a,b} S_{a,b}^- \exp(K_{a,b}) \right) \]

- Calculate kernel similarity matrix \( K \)

- At each stage of boosting,
  - Use loss to calculate a new data representation that will allow the algo to better satisfy the constraints on which it is performing poorly
  - Use eigen decomposition, find greatest inconsistencies
  - Project data onto new space
  - Cluster in new space; get pairwise clusterings
  - Compute performance and update \( K \) accordingly
  - Repeat until either all constraints satisfied or convergence

- Eigen decomp on \( K \), cluster with algo, return pairwise clusterings
Results

- BoostCluster is consistent: ave accuracy very close to max accuracy
- BoostKmeans < Spectral < BoostSpectral
- BoostCluster with spectral algo kicks ass!