A Framework for Feature Selection in Clustering

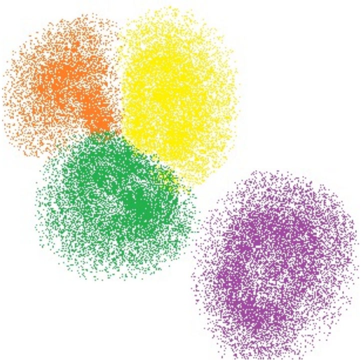
Daniela M. Witten and Robert Tibshirani

Stanford University

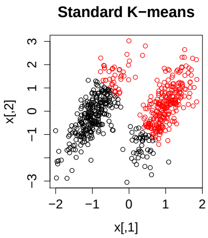
Wentao Wu

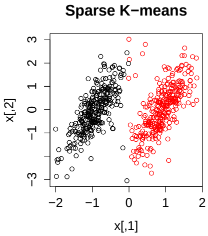
Problem

- Clustering: Given $X_{n \times p}$, identify K clusters.
- Sparse Clustering: Underlying clusters only differ on $q < p$ features.
  - improve accuracy and interpretation
  - cheaper prediction



Motivating Example





$$X_1 \sim N\left[\begin{pmatrix} -0.5 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.78 \\ 0.78 & 1 \end{pmatrix}\right]$$

$$X_2 \sim N\left[\begin{pmatrix} 0.5 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.78 \\ 0.78 & 1 \end{pmatrix}\right]$$

```r
#K-means
cl <- kmeans(x,2)
plot(x,col=cl$cluster, main='Standard K-
     means')
#Sparse K-means
km.perm <- KMeansSparseCluster.permute(x,K
     =2,wbounds=seq(3,7,len=15),nperms=5)
km.out <- KMeansSparseCluster(x,K=2,wbounds=
     km.perm$bestw)
plot(x,col=km.out[[1]]$Cs, main='Sparse K-
     means')
```

Sparse Clustering

$$\begin{aligned} \max_{\Theta \in D} \quad & \sum_{j=1}^{p} w_j f_j(X_j, \Theta) \\ \text{subject to} \quad & \|w\|^2 \leq 1, \ \|w\|_1 \leq s, \ w_j \geq 0 \end{aligned}$$

- w_j is a weight corresponding to feature j, also indicates contribution.
- $w_1 = w_2 = \ldots = w_p$, reduces to the traditional clustering.
- L_1 penalty results in sparsity.
- L_2 penalty avoids trivial solution, (at most one element of w is nonzero).

How to optimize

- Alternating Iterative Algorithm
- Holding w fixed, optimize with respect to Θ.
  - Standard clustering procedure to a weighted version of the data.
- Holding Θ fixed, optimize with respect to w.
  - $$\begin{array}{ll} \max_{w} & w^T a \\ \text{subject to} & \|w\|^2 \leq 1,\ \|w\|_1 \leq s,\ w_j \geq 0 \end{array}$$

How to optimize

$$\max_{w} \quad w^T a$$
$$\text{subject to} \quad \|w\|^2 \le 1,\ \|w\|_1 \le s,\ w_j \ge 0$$

Introducing Lagrangian multipliers λ_1 for $\|w\|^2 - 1 \le 0$, λ_2 for $\|w\|_1 - s \le 0$ and μ_j for $w_j \ge 0$. We obtain the KKT condition:

$$\begin{aligned} w^T w - 1 \le 0 & \qquad \sum w_j - s \le 0 \\ \mu_j w_j \le 0 & \qquad w_j \ge 0 \\ \lambda_1(w^T w - 1) = 0 & \qquad \lambda_2(\sum w_j - s) = 0 \\ \mu_j w_j = 0 & \qquad -a + \lambda_1 w_j + \lambda_2 - \mu_j = 0 \end{aligned}$$

How to optimize

Based on the last equation, $\mu_j = -a + \lambda_1 w_j + \lambda_2$. Thus, $(-a + \lambda_2 + \lambda_1 w_j) w_j = 0$.

- Case 1: $\sum w_j - s < 0$. Thus, $\lambda_2 = 0$. If $a < 0$, then $w_j = 0$. Otherwise, $w_j = \frac{a}{\lambda_1}$, where λ_1 make sure that $w^T w = 1$.
- Case 2: $\sum w_j - s = 0$. Thus, $\lambda_2 \geq 0$. If $a < 0$, then $w_j = 0$. Otherwise, $w_j = \frac{a - \lambda_2}{\lambda_1}$, where λ_1 ensures that $w^T w = 1$ and λ_2 ensures $\sum w_j - s = 0$.

In summary,

$$w = \frac{S(a_+, \Delta)}{\|S(a_+, \Delta)\|^2}$$

- $a = f_j(X_j, \Theta)$.
- $S(x, c) = \text{sign}(x)(|x| - c)_+$.
- $\Delta = 0$ if that results in $\|w\|_1 \leq s$, otherwise, $\Delta > 0$ is chosen to yield $\|w\|_1 = s$.

How to optimize

```
FindDelta <- function(a,l1bound){
  #Find optimal delta which statisfy w constrains

  #if already statisfies, return 0
  if(l2n(a)==0 || sum(abs(a/l2n(a)))<=l1bound) return(0)
  #binary search lo is 0, hi is maximum absolute element of a
  delta.lo <- 0
  delta.hi <- max(abs(a))-1e-5
  iter <- 1
  while(iter<=15 && (delta.hi-delta.lo)>(1e-4)){
    #cal S(a,delta)
    su <- soft(a,(delta.lo+delta.hi)/2)
    if(sum(abs(su/l2n(su)))<l1bound){
      #if stasifies the l1 bound, try a smaller delta
      delta.hi <- (delta.lo+delta.hi)/2
    } else {
      #if not, try last delta
      delta.lo <- (delta.lo+delta.hi)/2
    }
    iter <- iter+1
  }
  return((delta.lo+delta.hi)/2)
}
```

Sparse K-Means

$$\max_{C_1,C_2,\ldots,C_K,w} \quad \sum_{j=1}^{p} w_j\left(\frac{1}{n}\sum_{i=1}^{n}\sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K}\frac{1}{n_k}\sum_{i,i'\in C_k} d_{i,i',j}\right)$$

$$\text{subject to} \quad \|w\|^2 \leq 1, \quad \|w\|_1 \leq s, \quad w_j \geq 0$$

- w is the weight for each feature.
- K is the number of clusters.
- n_k is the number of observations in cluster k.
- C_k contains the indices of the observations in cluster k.
- $d_{i,i',j}$ is the dissimilarity measure between observation i and i' along feature j.

Optimize Sparse K-Means Clustering

1. Initialize w as $w_i = \frac{1}{\sqrt{p}}$, where $i = 1, \ldots, p$.
2. Iterative until convergence $\frac{\sum_{j=1}^{p} |w_j^r - w_j^{r-1}|}{\sum_{j} = 1^p |w_j^{r-1}|} < 10^{-4}$.
    1. Holding w fixed, optimize with respect to $C_1, \ldots, C_K$. Applying the standard K-means algorithm to the $n \times n$ dissimilarity matrix with (i, i') element $\sum_j w_j d_{i,i',j}$.
    2. Holding $C_1, \ldots, C_K$ fixed, optimize with respect to w by applying

    $$w = \frac{S(a_+, \Delta)}{\|S(a_+, \Delta)\|^2}$$

    where $a_j = \frac{1}{n} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_{i,i',j} - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,i' \in C_k} d_{i,i',j}$
3. The clusters are given by $C_1, \ldots, C_K$ and the feature weights corresponding to this clustering are given by $w_1, \ldots, w_p$.

Optimization

```r
UpdateC <- function(x, K, w, C=NULL){
  #num of row
  n <- nrow(x)
  #remove zero-weight features
  x <- x[,w!=0];
  xr <- sweep(x, 2, sqrt(w[w!=0]), "*");
  if(!is.null(C)){
    #num of colomn
    p <- ncol(x)
    #attach cluster information
    xr <- data.frame(data=xr, clu=C)
    #calculate cluster mean
    mu <- aggregate(xr,by=list(xr$clu),FUN=mean)
    #cal new dist including the cluster mean
    xr <- as.matrix(xr[,1:p])
    mu <- mu[,1:p+1]
    mu <- as.matrix(mu)
    distmat <- as.matrix(dist(rbind(xr, mu)))[1:n, (n+1):(n+K)]
    #get the cluster
    c_n <- apply(distmat, 1, which.min)
    #do the clustering
    if(length(unique(c_n))==K){
      kc <- kmeans(xr,centers=mu)
    }else{
      kc <- kmeans(xr,centers=K, nstart=10)
    }
  }else{
    kc <- kmeans(xr, centers=K, nstart=10)
  }
  return(kc$cluster)
```

Optimization

```r
UpdateW <- function(x, C, l1bound){
  #calculate WCSS
  wcss.feature <- CalWCSS(x, C)$wcss.feature
  #calculate TSS
  tss.feature <- CalWCSS(x, rep(1, nrow(x)))$wcss.feature
  #find delta
  delta <- FindDelta(-wcss.feature+tss.feature, l1bound)
  #soft operation
  wu.unscaled <- soft(-wcss.feature+tss.feature,delta)
  return(wu.unscaled/l2n(wu.unscaled))
}
```

Tuning Parameter

- s the L_1 bound on w.
- Use gap statistics.
- Gap statistic measures the strength of the clustering obtained on the real data relative to the clustering obtained on null data that does not contain subgroups.

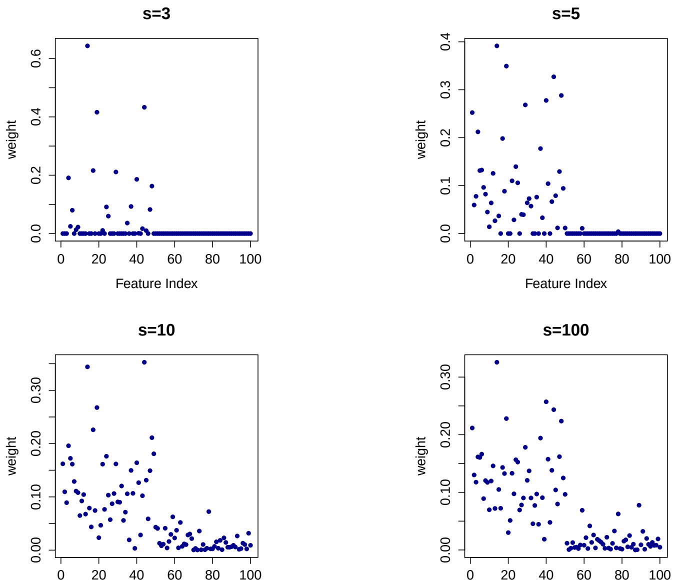
Tuning Parameter

1. Obtain permuted datasets $X_1, \ldots, X_B$ by independently permuting the observations within each feature.
2. For each candidate tuning parameter value s:
   1. Compute $O(s)$ the objective obtained by performing sparse K-means with tuning parameter s on the data X.
   2. For $b = 1, \ldots, B$, compute $O_b(s)$.
   3. Calculate $gap(s) = \log\ (O(s)) - \frac{1}{B}\sum_{b=1}^{B} \log\ (O_b(s))$.
3. Choose s^* as the smallest value for which $gap(s^*)$ is within a standard deviation of the largest value of $gap(s)$.

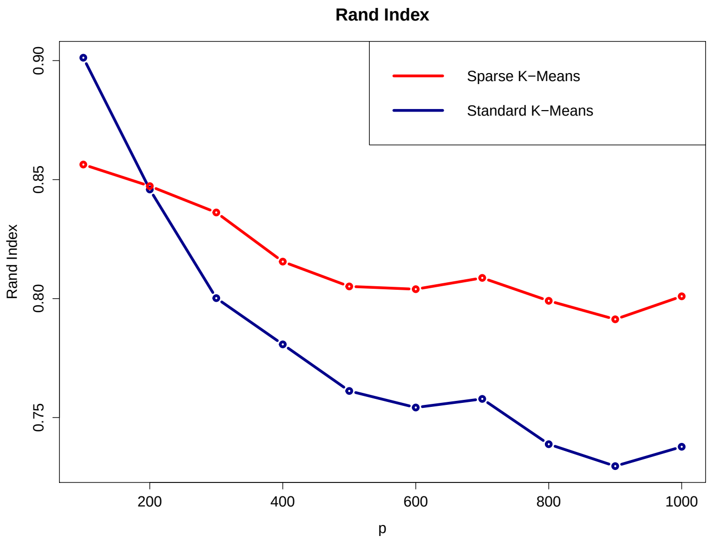
Simulation

- Three Clusters: C_1, C_2, C_3.
- Each cluster contains 20 observations.
- $p = 100, q = 50$: 100 features, the underlying clusters depend on the first 50 features.
- $X_{ij} \sim N(\mu_{ij}, 1)$.
- If $i \in C_k$ and $j \leq q$, $\mu_{ij} = \mu_{c_k}$, where $\mu_{c_1} = 0, \mu_{c_2} = 0.6, \mu_{c_3} = 1.2$.
- If $j > q$, $\mu_{ij} = 0$ regardless of i.
- Metric: Rand index

Simulation-Different S

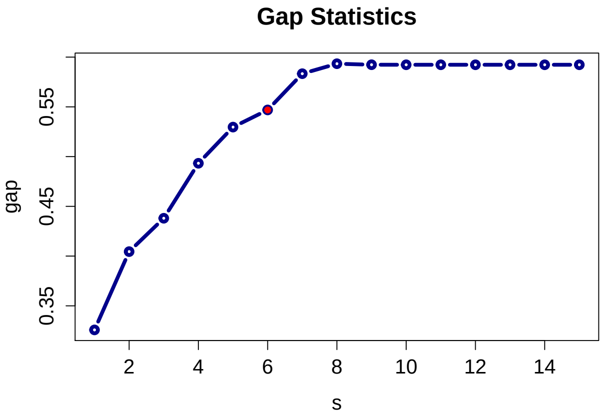


Simulation-Different p



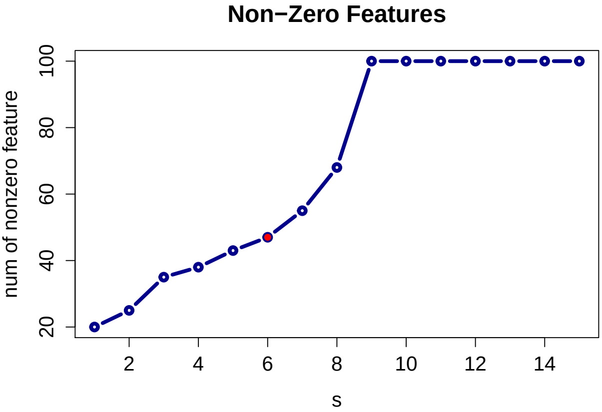
Simulation-Tuning

```r
k.perm <- SparseKMeansTuning(x,K=3,wbounds=seq(3,10,len=15), nperm=10)
```



Simulation-Tuning

```
k.perm <- SparseKMeansTuning(x,K=3,wbounds=seq(3,10,len=15), nperm=10)
```



Sparse Hierarchical

$$\max_{U} \quad \{\sum_j \sum_{i,i'} w_j d_{i,i',j} U_{i,i'}\}$$
$$\text{subject to} \quad \sum_{i,i'} U_{i,i'}^2 \leq 1,\ \|w\|^2 \leq 1,\ \|w\|_1 \leq s,\ w_j \geq 0$$

- $U_{i,i'} \propto \sum_j w_j d_{i,i'j}$.
- Performing hierarchical clustering on U results in the sparse hierarchical clustering.

How to optimize

1. Initialize w as $w_1 = \ldots = w_p = \frac{1}{\sqrt{p}}$.
2. Iterate until convergence:
    1. Update $u = \frac{Dw}{\|Dw\|_2}$.
    2. Update $w = \frac{S(a_+, \Delta)}{\|S(a_+, \Delta)\|_2}$ where $a = D^T u$ and $\Delta = 0$ if this results in $\|w\|_1 \leq s$; otherwise, $\Delta > 0$ is chosen such that $\|w\|_1 = s$.
3. Rewrite u as a $n \times n$ matrix U.
4. Perform hierarchical clustering on the $n \times n$ dissimilarity matrix U.

The End