Machine Learning for OR & FE

Unsupervised Learning: Clustering

Martin Haugh

Department of Industrial Engineering and Operations Research
Columbia University
Email: `martin.b.haugh@gmail.com`

(Some material in these slides was freely taken from **Garud Iyengar's** slides on the same topic.)

Outline

Supervised Versus Unsupervised Learning

The goal of supervised learning is to predict an output y as a function of inputs $\mathbf{x} = (x_1, \ldots, x_p)$

- We are given training data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
  - $(\mathbf{x}_i, y_i)$ is known for all $i = 1, \ldots, N$.
- The goal is to compute $\mathsf{E}[y \mid x]$ or otherwise predict y given $\mathbf{x}$.

Regression and classification are examples of supervised learning.

The goal of unsupervised learning is to characterize the distribution $\mathbb{P}(\mathbf{x})$ of the inputs $\mathbf{x}$. There is no target output y.

Standard examples of unsupervised learning methods include:

- Clustering methods: Identify multiple regions of the $\mathbf{x}$-space that contain modes of $\mathbb{P}(\mathbf{x})$. One can then represent $\mathbb{P}(\mathbf{x})$ as a mixture of simpler densities representing distinct types or classes of observations.
- Dimension-reduction methods: Identify low-dimensional manifolds in $\mathbf{x}$-space that represent high data density. Examples include principal component analysis (PCA), independent component analysis (ICA), spectral clustering, etc.

Clustering Methods

The goal with clustering methods is to partition the data into clusters with low intra-cluster dissimilarity and large inter-cluster dissimilarity.

There are many approaches:

- Non-probabilistic combinatorial methods: K-means, K-medoids etc.
- Mixture modeling: Data is an IID draw from a mixture but the mixture indicator is latent or hidden.
- Mode-seeker: Seek modes of the joint PDF of $\mathbf{x}$ in a non-parametric manner

We will focus on the first method here but will consider mixture modeling when we study the EM algorithm.

Object Dissimilarity

A key component of any clustering algorithm is how the distance, $d(\mathbf{x}, \mathbf{y})$, between points $\mathbf{x}$ and $\mathbf{y}$ are measured.

- Will always assume $d(\mathbf{x}, \mathbf{x}) = 0$
- Typically assume $d(\cdot, \cdot)$ is symmetric: $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
- In general do not assume the triangle inequality holds
  - so there may exist $\mathbf{x}, \mathbf{y}, \mathbf{z}$ such that $d(\mathbf{x}, \mathbf{z}) > d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.

Examples:

- Weighted difference between attributes: $d(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{p} w_j d_j(x_j, y_j)$
- Quantitative variables: $d_j(x_j, y_j) = f_j(|x_j - y_j|)$, for $f_j(\cdot)$ non-negative and non-decreasing.
- Rank variables: the rank $i \mapsto \frac{i-1/2}{N}$, $i = 1, \ldots, N$
  - now treat as quantitative variable.
- Categorical variables: often choose $d(x, y) := c\,1_{\{x \neq y\}}$.

Dissimilarity matrix: $\mathbf{D} \in \mathbb{R}^{N \times N}$ with $D_{ij} := d(\mathbf{x}_i, \mathbf{x}_j)$

– if clustering algorithm requires symmetric $\mathbf{D}$ then can use $(\mathbf{D} + \mathbf{D}')/2$.

How should we handle missing data?

How Do We Choose the w_j's?

We have N points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ so that $d_k(x_{ik}, x_{jk}) =$ dissimilarity measure on the k^{th} attribute of $\mathbf{x}_i$ and $\mathbf{x}_j$.

Combined dissimilarity measure: $d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} w_k d_k(x_{ik}, x_{jk})$

How do we set the weights w_k? Total average dissimilarity is:

$$\bar{d} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} w_k \cdot \underbrace{\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d_k(x_{ik}, x_{jk})}_{\bar{d}_k}$$

In order to treat all attributes equally we should choose: $w_k \sim \frac{1}{\bar{d}_k}$

– and also typically have $\sum_{k=1}^{p} w_k = 1$.

For $d_k(x_{ik}, x_{jk}) = (x_{ik} - x_{jk})^2$, the normalization is $\bar{d}_k = 2\text{Var}(x_k)$

But equal weighting is not always appropriate

– see Figure 14.5 from HTF.

To Scale or Not to Scale?

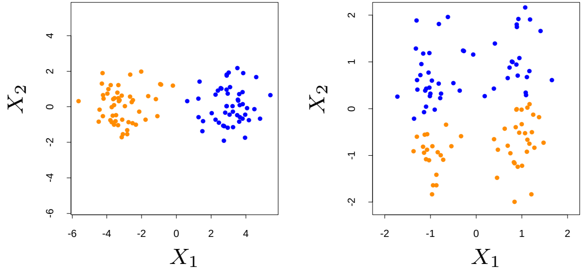


FIGURE 14.5. *Simulated data: on the left, K-means clustering (with $K = 2$) has been applied to the raw data. The two colors indicate the cluster memberships. On the right, the features were first standardized before clustering. This is equivalent to using feature weights* $1/[2 \cdot \mathrm{var}(X_j)]$. *The standardization has obscured the two well-separated groups. Note that each plot uses the same units in the horizontal and vertical axes.*

Figure 14.5 taken from HTF

Clustering Algorithms

A clustering algorithm with K clusters and $N \gg K$ points should produce:

- A cluster map $i \in \{1, \ldots, N\} \mapsto \{1, \ldots, K\}$.
  The k^{th} cluster is then $\mathcal{C}_k := \{i : i \mapsto k\}$
- Cluster "centers" $\mathbf{m}_k$ for $k = 1, \ldots, K$
  - $\mathbf{m}_k$ approximates all points in the k^{th} cluster.

Within and between segment dissimilarity can be computed as

$$\begin{aligned}
\frac{1}{2}\sum_{i=1}^{N}\sum_{\ell=1}^{N} d(\mathbf{x}_i, \mathbf{x}_\ell) &= \frac{1}{2}\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\left(\sum_{\ell\in\mathcal{C}_k} d(\mathbf{x}_i, \mathbf{x}_\ell) + \sum_{\ell\notin\mathcal{C}_k} d(\mathbf{x}_i, \mathbf{x}_\ell)\right) \\
&= \frac{1}{2}\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\sum_{\ell\in\mathcal{C}_k} d(\mathbf{x}_i, \mathbf{x}_\ell) + \frac{1}{2}\sum_{k=1}^{K}\sum_{i\in\mathcal{C}_k}\sum_{\ell\notin\mathcal{C}_k} d(\mathbf{x}_i, \mathbf{x}_\ell) \\
&= W(\mathcal{C}) + B(\mathcal{C})
\end{aligned}$$

$W(\mathcal{C})$ = within-cluster dissimilarity and $B(\mathcal{C})$ = between-cluster dissimilarity.

Clustering Algorithms

Goal: Find clusters $\mathcal{C} := \{\mathcal{C}_k : k = 1, \dots K\}$ so that within-cluster dissimilarity is minimized

– or equivalently (why?) so that between cluster dissimilarity is maximized.

But this is an NP-hard problem!

It can be shown that the number of possible cluster assignments is

$$S(N, K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} k^N$$

According to HTF, $S(10, 4) = 34,105$ and $S(19, 4) \approx 10^{10}$

– so $S(N, K)$ grows rapidly and there is no chance of using enumeration to find global optimal for reasonable values of (N, K).

Instead we seek a good local optimum

– typically using algorithms based on iterative greedy descent.

K-Means Clustering

K-means is one of the most popular clustering algorithms.

The dissimilarity measure is squared Euclidean distance: $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$

– so all variables required to be of the quantitative type.

For each $\mathcal{C}_k$ the minimum distortion center, $\mathbf{m}_k$, is easily found ...

$$\mathbf{m}_k := \underset{\mathbf{m}}{\operatorname{argmin}} \sum_{\mathbf{x}_i \in \mathcal{C}_k} \|\mathbf{x}_i - \mathbf{m}\|_2^2 = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} \mathbf{x}_i$$

Moreover can easily check

$$W(\mathcal{C}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i,j \in \mathcal{C}_k} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \sum_{k=1}^{K} N_k \sum_{i \in \mathcal{C}_k} \|\mathbf{x}_i - \mathbf{m}_k\|_2^2 \, .$$

K-Means Clustering

The K-means clustering problem is to solve:

$$\min_{\mathcal{C}} \min_{\{\mathbf{m}_k\}_{k=1}^K} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\mathbf{x}_i - \mathbf{m}_k\|_2^2$$

As explained earlier, we cannot hope to find the global optimal to this problem.

So we use the following iterative algorithm:

Step 1. Fix clusters $\{\mathcal{C}_k\}$ and set $\mathbf{m}_k = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} \mathbf{x}_i$

Step 2. Fix $\{\mathbf{m}_k\}$ and choose clusters: $\mathcal{C}(i) = \mathrm{argmin}_k \{\|\mathbf{x}_i - \mathbf{m}_k\|_2\}$

– so clusters $\{\mathcal{C}_k\}$ are identified in this step.

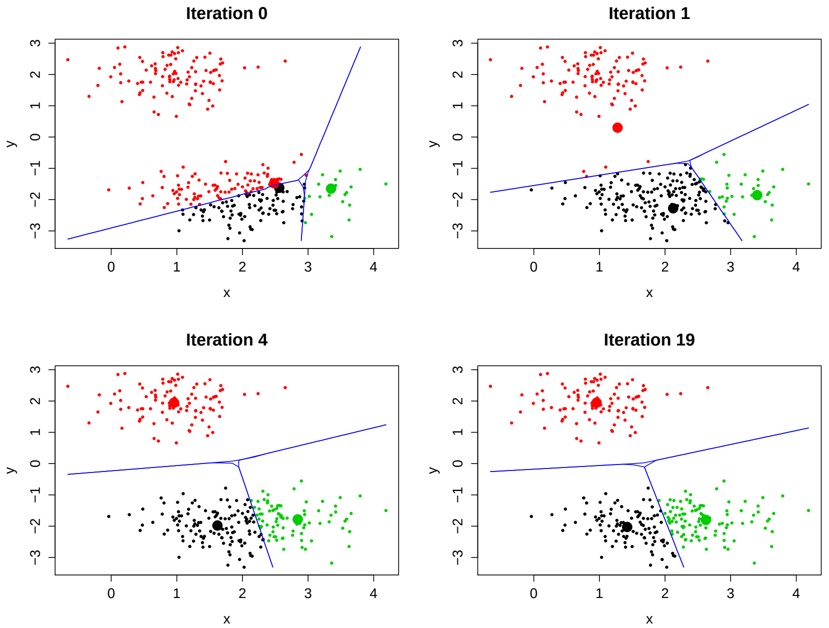
Some approximation guarantees are available for this algorithm.

And it is guaranteed (why?) to converge to a local minimum

– so always a good idea to run algorithm from many different starting points.

Next slide shows snapshots of the K-means iterations

– clusters are given by the Voronoi regions determined by the blue lines.


Iteration 0
Iteration 1
Iteration 4
Iteration 19
x
y
0
1
2
3
4
−3
−2
−1
0
1
2
3

Application: Vector Quantization

Note that a gray-scale image can be represented as a vector $\mathbf{X} \in \mathbb{R}^n$

- e.g. the left image of statistician Ronald Fisher consists of $1,024 \times 1,024$ pixels and can therefore be represented by $\mathbf{X} \in \mathbb{R}^{1024 \times 1024}$
- each pixel is a grayscale value ranging from 0 to 255
  - so each pixel requires 1 byte of storage
- so entire image requires approx 1 megabyte of storage.

For an image $\mathbf{X}$, can define a collection of vectors $\mathcal{X} := \{\mathbf{x}_{ij}\} \subset \mathbb{R}^4$ with

$$\mathbf{x}_{ij} := \mathbf{vec}\left(\begin{bmatrix} X_{2i-1,2j-1} & X_{2i,2j-1} \\ X_{2i-1,2j} & X_{2i,2j} \end{bmatrix}\right)$$

Can now define each $\mathbf{x}_{ij}$ as a data-point and use K-means clustering to compress the image.

Application: Vector Quantization

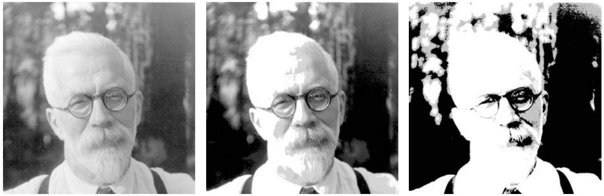


FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2×2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

Figure 14.9 taken from HTF

Application: Vector Quantization

So run K-means on $\mathcal{X}$ to obtain cluster “centers” $\{\mathbf{m}_k\}_{k=1}^K \subseteq \mathbb{R}^4$

Each group of 4 pixels is replaced by its assigned cluster center

- so if $\mathcal{C}(\mathbf{x}_{ij}) = k$ we then set

$$\begin{bmatrix} \hat{X}_{2i-1,2j-1} & \hat{X}_{2i,2j-1} \\ \hat{X}_{2i-1,2j} & \hat{X}_{2i,2j} \end{bmatrix} = \mathbf{mat}(\mathbf{m}_k)$$

where $\mathbf{mat}(\mathbf{m}_k)$ is the 2×2 matrix constructed from the 4×1 vector, $\mathbf{m}_k$.

This compression algorithm is known as lossy compression.

Question: How do we use the output of the K-means algorithm to compress the image?

Question: How would we vary the amount of compression?

Example: Clustering of Senators

Roll call data is a great resource for measuring "dissimilarity" between legislators.

Keith Poole (University of Georgia) and Howard Rosenthal (NYU) maintain an interesting web-site on this topic at http://www.voteview.com.

Question: How could we cluster senators?

Answer: We need a measure of measure of dissimilarity between senators.

Let $\mathbf{V} = [v_{ik}]$ denote the votes of legislator i on bill k

$$v_{ik} = \begin{cases} +1 & \text{vote of legislator } i \text{ on bill } k \text{ is ``aye''} \\ -1 & \text{vote of legislator } i \text{ on bill } k \text{ is ``nay''} \\ 0 & \text{legislator } k \text{ abstained on bill } k \end{cases}$$

Dissimilarity between legislators $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_k (v_{ik} - v_{jk})^2}$.

Is this a reasonable dissimilarity measure?

– if so, then we can just go ahead and run K-means.

Kernel K-Means Clustering

Can extend K-means to general kernels $K(\mathbf{x}, \mathbf{y}) \neq \mathbf{x}^\top \mathbf{y}$ via the "kernel trick".

Mercer's Theorem implies $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$ for some (possibly unknown) $\phi : \mathbb{R}^p \mapsto \mathbb{R}^d$ where typically $d \gg p$.

Recall that the cluster centers satisfy

$$\mathbf{m}_k := \operatorname{argmin} \sum_{i \in \mathcal{C}_k} \|\phi(\mathbf{x}_i) - \mathbf{m}_k\|_2^2 = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} \phi(\mathbf{x}_i)$$

Cluster assignment in iteration $t+1$

$$\begin{aligned} \mathcal{C}^{(t+1)}(i) &= \operatorname*{argmin}_k \|\phi(\mathbf{x}_i) - \mathbf{m}_k\|_2^2 \\ &= \operatorname*{argmin}_k \left\{ K(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{N_k} \sum_{\ell \in \mathcal{C}_k^{(t)}} K(\mathbf{x}_i, \mathbf{x}_\ell) + \frac{1}{N_k^2} \sum_{\ell, \ell' \in \mathcal{C}_k^{(t)}} K(\mathbf{x}_\ell, \mathbf{x}_{\ell'}) \right\} \end{aligned}$$

So the cluster assignment can be computed without reference to ϕ

– only the kernel function $K(\cdot, \cdot)$ is required!

Question: But what about computing the $\mathbf{m}_k$'s?

K-Medoids Clustering

K-means is very sensitive to outliers as they produce large distances

– so maybe worth exploring other distance measures.

Fact: Given $\mathbf{x} = \{x_1, \ldots, x_N\}$, median$(\mathbf{x}) \in \operatorname{argmin}_z \sum_{i=1}^N |x_i - z|$

– so could cluster by using the median to construct distance measure.

More generally, we can assign cluster "center" to be the cluster medoid so we set $\mathbf{m}_k := \mathbf{x}_{i_k}$ where

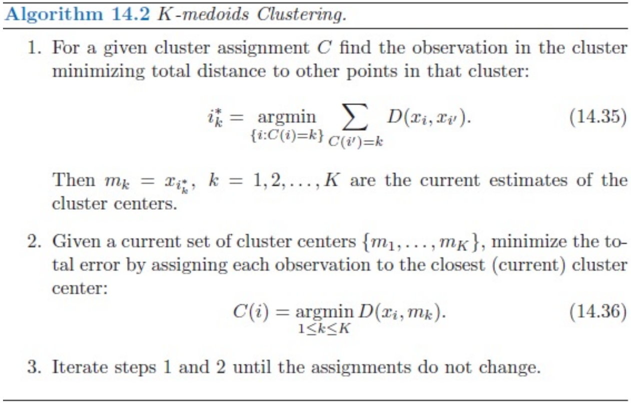
$$i_k = \underset{\ell \in \mathcal{C}_k}{\operatorname{argmin}} \sum_{i \in \mathcal{C}_k} d(\mathbf{x}_\ell, \mathbf{x}_i).$$

Form new clusters by assigning points to the nearest cluster center

$$\mathcal{C}(i) = \underset{1 \le k \le K}{\operatorname{argmin}}\, d(\mathbf{x}_i, \mathbf{m}_k)$$

This yields the K-medoids clustering algorithm ...

K-Medoids Clustering



Algorithm 14.2 *K-medoids Clustering.*

1. For a given cluster assignment C find the observation in the cluster minimizing total distance to other points in that cluster:

$$i_k^* = \underset{\{i:C(i)=k\}}{\operatorname{argmin}} \sum_{C(i')=k} D(x_i, x_{i'}). \tag{14.35}$$

   Then $m_k = x_{i_k^*}$, $k = 1, 2, \ldots, K$ are the current estimates of the cluster centers.

2. Given a current set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} D(x_i, m_k). \tag{14.36}$$

3. Iterate steps 1 and 2 until the assignments do not change.

Algorithm 14.2 in HTF.

Note that we do not need to explicitly compute cluster centers

– so only dissimilarity matrices rather than actual points are needed.

But K-medoids is more computationally demanding than K-means.

How to Choose the Number of Clusters K

A key practical issue is how to choose the number of clusters, K.

Sometimes the appropriate value of K is obvious from domain specific knowledge

– e.g. suppose a company has K sales-people so the goal is to partition the customer database into K segments.

But often we need to use the data to determine an appropriate value of K.

So let $W_K(\mathcal{C})$ denote within-cluster-dissimilarity when we have K clusters.

Then we expect $W_K(\mathcal{C}) > W_{K+1}(\mathcal{C})$ for any value of K but if K^* is the "correct" number of clusters then should have

$$\Delta W_{K^*}(\mathcal{C}) \gg \Delta W_{K^*+1}(\mathcal{C})$$

– so a heuristic approach is to look for a kink in the plot of $W_K(\mathcal{C})$ versus K

– and choose K^* to be the point at which the kink occurs.

HTF also discuss the gap statistic which compares the curve $\log(W_K(\mathcal{C}))$ to the curve obtained from uniformly distributed data over a rectangle containing the original data

– can automate the choice of K^* using this statistic

– see Section 14.3.11 of HTF for further details.

Hierarchical Clustering

Hierarchical Clustering maintains a hierarchy of clusters whereby:

- Clusters at a given level are created by merging clusters at the next lower level
- Each cluster at the lowest level contains a single observation.
- At the highest level there is just one cluster containing all the data.

There are two basic paradigms for constructing hierarchical clusters:

- Agglomerative: bottom-up
- Divisive: top-down

Methods for agglomerative clustering: merge clusters $\{G, H\}$ with minimum

- Single linkage (SL) distance: $d_{SL}(G, H) = \min_{i \in G, j \in H} D_{ij}$
- Complete linkage (CL) distance: $d_{CL}(G, H) = \max_{i \in G, j \in H} D_{ij}$
- Group average (GA) distance: $d_{GA}(G, H) = \frac{1}{N_H N_G} \sum_{i \in G, j \in H} D_{ij}$

Normal Mixture Models

Clustering via normal mixture models is an example of probabilistic clustering

- we assume the data are IID draws.

e.g. (Scalar Case) Suppose $\mathcal{X} = (X_1, \ldots, X_n)$ are IID random variables each with PDF

$$f_x(x) \;=\; \sum_{j=1}^{m} p_j \, \frac{e^{-(x-\mu_j)^2/2\sigma_j^2}}{\sqrt{2\pi\sigma_j^2}}$$

where $p_j \geq 0$ for all j and where $\sum p_j = 1$

- parameters are the p_j's, the μ_j's and the σ_j's
- typically estimated via MLE
- which we can do via the EM algorithm
    - an extremely important algorithm in statistics / machine learning.

We will study the EM algorithm as a separate topic

- will then see clustering via normal mixture models as an application of EM.

A Detour: Multidimensional Scaling

Let $\mathbf{D}$ denote a dissimilarity matrix for the data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

– the underlying data might be categorical, network, or text.

Want to embed data in $\mathbb{R}^k$ where $k \leq n - 1$

– goal is to "visualize" the data so default is $k = 2$.

Compute locations $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^2$ so that $\left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2 \approx d(\mathbf{x}_i, \mathbf{x}_j)$

– i.e. want an embedding that preserves "distances".

Can be formulated as a multidimensional scaling problem:

$$\min_{\mathbf{y}_1,\ldots,\mathbf{y}_N} S_M(\mathbf{y}_1, \ldots, \mathbf{y}_N) := \sum_{i \neq j} \left(d(\mathbf{x}_i, \mathbf{x}_j) - ||\mathbf{y}_i - \mathbf{y}_j||\right)^2$$

– there are other similar formulations; see Section 14.8 of HTF.

Simple example

Respondents rate products and yield the Response matrix:

	P1	P2	P3	P4	P5	P6
A	0	-1	0	-1	0	0
B	-1	0	1	1	1	0
C	0	0	0	1	-1	1
D	1	0	1	-1	0	0

Want to visualize the clustering of the respondents.

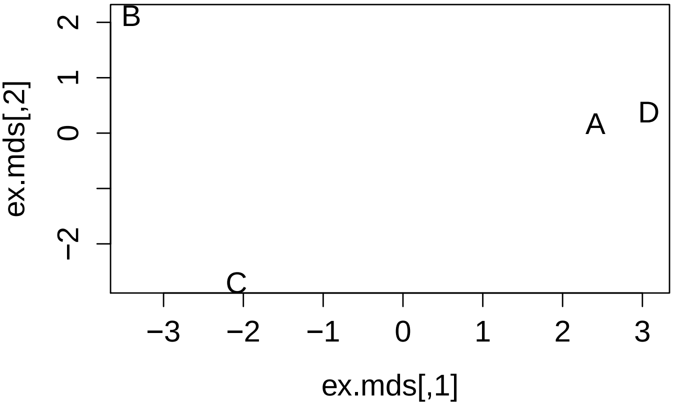
- Dissimilarity: $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{4} |x_{ki} - x_{kj}|^2}$ (R command: `dist`)

  where $\mathbf{X} :=$ Response $\times$ Response$^\top$

	A	B	C
B	6.24		
C	5.48	5.00	
D	2.24	6.78	6.08

- Embed dissimilarity matrix in $\mathbb{R}^2$ (R command: `cmdscale`)

Simple example (contd)


B
A
D
C
ex.mds[,2]
ex.mds[,1]