

---

# Orthogonal clustering using predictability

---

Abhishek Gupta

Dean P. Foster

Department of Statistics, University of Pennsylvania, Philadelphia, PA 19104 USA

ABGUPTA@WHARTON.UPENN.EDU

FOSTER@WHARTON.UPENN.EDU

Lyle H. Ungar

Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA

UNGAR@CIS.UPENN.EDU

## Abstract

Faceted classification is a flexible way of organizing information based on multiple independent labelings. Faceting is usually contrasted with clustering. Both have similar objectives of organizing data, but while clustering does this in an automated way, faceting requires considerable manual labeling. In this paper we aim to solve the problem of generating these facets automatically. We propose a modification of the CPCM algorithm, the Ortho-CPCM to solve this orthogonal clustering problem. The resulting clusters inherit properties of CPCM, namely invariance to linear transformations and tend to eliminate noise features by driving their weights to zero.

## 1. Introduction

*Faceted Classification* is an increasingly common method to organize information in the digital age. Faceted classification gives multiple labels to an object and each of these labels are from mutually-exclusive categories. Using these labels, a user can navigate an information hierarchy in the manner he wants to e.g. books can be given labels according to author, subject type, cover (hard vs paperback), language, and so on. Each of these categories are mutually exclusive and a user may be interested to get results in any order of classification. These classifications have been traditionally used in cataloging and in building faceted navigation systems. An example of such a system is the *Flamenco* search interface (Yee et al., 2003). Facets have proven in

some user studies to give superior user experience. Faceting also has found use in software reuse. Figure 1 shows an example of a facet scheme for unix tools.

A natural association of faceting can also be made to classification trees. But it is important to note that if this hierarchy is viewed as a tree, then at nodes on the same level we have the exact same splits. Thus each level corresponds to the application of a clustering function on the same direction. Also the order of the levels (and hence the application of clustering functions) does not matter. Faceting is a hierarchical representation which is very flexible.

Faceting is usually contrasted with clustering (Hearst, 2006). Both these methods aim to group objects in a meaningful way. Clusters have the virtue of being automatically constructed, usually with the help of user-input similarity metrics. In fact clustering is an important problem in machine learning with ubiquitous applications. This automatic grouping is also viewed by many as a loss of flexibility in browsing. The user gives up the control, to explore data in possibly alternate ways, to an algorithm. The fact that most things can be classified in multiple ways, makes a more flexible representation possible by faceted classification. In this paper we try to solve the unsupervised version of *faceted* classification by proposing a method for automatically constructing facets.

The aim here is to generate facets automatically by using clustering. The main idea of the paper is to use the *Clustering Predictions of Cluster Memberships* (CPCM) algorithm (Gupta et al., 2007). Each clustering obtained in CPCM is a projection to a line. Thus taking projections in orthogonal directions, we can get orthogonal clusterings. We call this Ortho-CPCM. This paper explains the orthogonal clustering and Ortho-CPCM in detail. We begin

DOMAIN -> UNIX tools			
(by action)	(by object)	(by data structure)	(by system)
get	file-names	buffer	line-editor
put	identifiers	tree	text-formater
update	line-number	table	
append	character	file	
check	number	archive	
detect	expression		
locate	entry		
search	declaration		
evaluate	line		
compare	pattern		
make			
build			
start			

Figure 1. An example of a faceted scheme for unix components. Figure from Prieto-Diaz (1990)

with an overview of the CPCM algorithm in the following section.

## 2. Learning Distance Metric using Predictability

This section follows (Gupta et al., 2007) which proposes a method of learning metrics in an unsupervised setting. A good distance metric would lead to tight and well-separated clusters in some projected space. This is quantified by introducing a new criterion, the ratio of the average (adjusted) distance of points to their nearest cluster centers to the average (adjusted) distance of the data points to their overall mean. For a linear transformation  $\mathbf{A}$  of the data, this criterion is called the *blur* ratio  $BR(\mathbf{A})$ . The goal then is to find the  $\mathbf{A}$  which minimizes this *blur* ratio. For this case the criterion resembles to the one that LDA minimizes, except that here it is unsupervised learning. In effect we are learning a distance metric and the transformation obtained projects the data to a subspace where the clusters are tight. Minimization of the blur-ratio handles data with high-dimensional noise features better.

For the minimization an iterative algorithm, “Clustering Predictions of Cluster Membership (CPCM),” is used which first predicts cluster membership, and then defines new clusters by clustering the predictions of cluster membership. The intuition behind using predictability is that if we generate clusters using a set of features, we should also be able to predict the membership of the clusters using the same features. By using predictions of cluster membership, we can take advantage of supervised learning methods to better solve unsupervised and semi-supervised problems (Chapelle et al., 2003). In

CPCM a hard clustering algorithm is combined with a soft prediction algorithm (i.e. the prediction step predicts cluster probabilities and not cluster memberships).

### 2.1. Optimal Distance Metric and Blur Ratio

Consider a  $N \times p$  dimensional matrix  $\mathbf{X}$  where  $N$  is the number of points and  $p$  is the number of features. We will use  $\mathbf{X}_i$  to denote the  $i$ th row and  $\mathbf{X}_{\cdot i}$  to denote the  $i$ th column in  $\mathbf{X}$ . This makes the feature vector into a row vector. We will use row vectors throughout. Define  $\mathbb{C} = \{C_1, \dots, C_K\}$  to be the set of clusters. Let  $\Delta_{\mathbb{C}}$  be the simplex over the  $K$  dimensions (the subspace of  $K$  dimensions such that every point looks like a probability, with components lying on  $(0,1)$  and summing to 1) with  $e_k \in \Delta_{\mathbb{C}}$  denoting a unit direction. Denoting the clustering function by  $c$ , we have the following map

$$\mathbf{X} \xrightarrow{\text{k-means}} c(\cdot) : \mathcal{R}^p \rightarrow \mathbb{C} \quad (1)$$

Using  $c(\cdot)$ , we define the matrix  $\mathbf{Z}$  such that  $\mathbf{Z}_i = e_{k:c(\mathbf{X}_i)=C_k}$ . Note that  $\mathbf{Z}_i$  is a  $K$  dimensional row vector on the simplex  $\Delta_{\mathbb{C}}$ .  $\Delta_{\mathbb{C}}$  can be regarded as a probability simplex for the cluster membership with  $\mathbf{Z}_{ik}$  equaling the probability of point  $i$  being in cluster  $k$ . Define  $\boldsymbol{\mu}_k \in \mathcal{R}^p$  as the center in the feature space for cluster  $C_k$ .

Our goal is to find the linear transformation of the data  $\mathbf{X}$  such that the distance metric  $d(x, y) = \sqrt{(x - y)\mathbf{A}(x - y)^T}$  gives the lowest “blur” ratio, which is defined as the ratio of the distances of points from their cluster centers to the distance of the points to the mean of the entire data set, both in the transformed space. The metric corresponds to transforming the points  $x$  by multiplying with the

$K$ -dimensional matrix  $\beta$ , where  $\beta\beta^T = \mathbf{A}$ .

We will build up the blur ratio by a sums of squares decomposition:

Within cluster variance,  $SSC$

$$\equiv \sum_{k=1}^K \sum_{i:c(\mathbf{X}_i)=C_k} (\mathbf{X}_i - \mu_k)\mathbf{A}(\mathbf{X}_i - \mu_k)^T$$

Total variance,  $SST$

$$\equiv \sum_{i=1}^N (\mathbf{X}_i - \mu)\mathbf{A}(\mathbf{X}_i - \mu)^T$$

Here  $\mu = \bar{\mathbf{X}}$  and  $\mathbf{A}$  is a symmetric positive semi-definite matrix. We then can define the *blur ratio* and the optimization problem as

$$\min_{\mathbf{A}, c} BR(\mathbf{A}, c) \equiv \frac{SSC}{SST}$$

Following the same argument as for  $k$ -means type algorithms (Peng & Xia, 2005), it is clear that optimizing the blur ratio is NP-hard. Thus we rely on the existence of good approximate clustering algorithms.

Given the transformation  $\mathbf{A}$ , clustering minimizes the numerator by finding  $c()$  and hence obtaining the optimal  $\mu_k$ . Since the denominator does not change, it is clear that  $BR$  will decrease. Given the cluster partition, the optimum  $\mathbf{A}$  matrix unfortunately will be of rank one (a similar property was pointed out by Xing et al., 2003) Instead we want to ensure that the transformation minimizes the distance between cluster centers and a  $K$ -dimensional simplex, while maintaining the simplex structure. We therefore add the following constraint ( $\forall i \neq j$ )  $(\mu_i - \mu_j)\mathbf{A}(\mu_i - \mu_j)^T = 2$  which prevents the centers of two different clusters from overlapping, and keeps the  $\mathbf{A}$  matrix from collapsing to a rank one matrix. Note that the RHS of the constraint just has to be any positive number (which we choose as 2).

Without loss of generality, under this constraint, we can take that there exists a decomposition of  $\mathbf{A} = \beta\beta^T$  (since  $\mathbf{A}$  is positive definite) so that,  $SSC = \sum_{k=1}^K \sum_{i:c(i)=C_k} (\mathbf{X}_i\beta - \theta_k)(\mathbf{X}_i\beta - \theta_k)^T$ . Thus given the clustering, the SSC can be minimized by minimizing the  $L_2$  distance shown above, which boils down to finding the optimal  $\beta$  and hence  $\mathbf{A}$ . Thus

we can consider  $\theta_k \in \Delta_{\mathbb{C}} \subset \mathcal{R}^K$  as the center in the prediction space for cluster  $C_k$ .

Finding the optimal  $BR$  can thus be seen as a two stage optimization procedure. We first find an optimal partitioning of the data using a clustering method. Then, we find the optimal  $\mathbf{A}$  which reduces the within cluster variance in the transformed space. These two steps can be iterated till a fixed point is reached. This insight forms the basis of the CPCM algorithm, which we introduce in the next section.

## 2.2. CPCM: An Iterative Clustering Prediction algorithm

The CPCM algorithm alternates between two stages - Clustering and Prediction, as shown below. In this paper we use linear regression as the prediction algorithm, and  $k$ -means as the clustering algorithm.

More formally, the cluster prediction model is given by

$$\mathbf{Z}_i = \mathbf{X}_i\beta + noise \quad (2)$$

where  $\beta \in \mathcal{R}^{p \times K}$ . Estimating the cluster memberships of each point by least squares gives a prediction  $\hat{\mathbf{Z}}_i$ . Since  $\hat{\mathbf{Z}}_i \in \mathcal{R}^K$  (instead of just a scalar as in the case of a usual regression setting), we run a regression on each of the  $K$  columns of  $\mathbf{Z}$  ( $N \times K$  matrix) to generate the predictions.

The points are then clustered in the cluster-prediction space, which is a linear transformation of the original points.

$$\hat{\mathbf{Z}} \xrightarrow{k\text{-means}} c'(\cdot) : \mathcal{R}^K \rightarrow \mathbb{C}' \quad (3)$$

The algorithm is then iterated.

## 2.3. The algorithm

A summary of the algorithm is given below.

---

### Algorithm 1 CPCM

---

**Input:** Data  $\mathbf{X}$ . Set  $t=0$

Generate an initial set of random clusters  $\mathbb{C}^{(0)}$

**repeat**

Predict cluster membership  $\mathbf{Z}_i^{(t)}$  based on  $\mathbf{X}$  and  $\mathbb{C}^{(t)}$

Generate  $\mathbb{C}^{(t+1)}$  by clustering  $\hat{\mathbf{Z}}^{(t)}$ .

Increment  $t$

**until**  $BlurRatio(t) = BlurRatio(t-1) + \epsilon$ .

---

Key to CPCM is the fact that it clusters in a simplex space. Suppose that we are seeking  $K$  clusters. We

can view these  $K$  clusters in a  $K$  dimensional simplex  $\Delta_{\mathbb{C}}$  which lies in a  $K - 1$  dimensional space, i.e the  $K$  vertices and all their interconnecting line segments, polygonal faces etc. lie in a  $K - 1$  dimensional space. The predicted cluster memberships can then be viewed as probabilities, which lie in this space.

### 3. Orthogonal Clustering

We now give two definitions of orthogonal clustering and show that an algorithm based on CPCM finds orthogonal clusters satisfying both the definitions. As in section 2, we consider  $\mathbf{X}$  to be the  $N \times p$  dimensional data matrix.

**Definition 1** *Orthogonal clustering corresponds to an orthogonal factorization of  $\mathbf{X} = \mathbf{H}\mathbf{B}^T$ , where  $\mathbf{H}$  is a  $N \times m$  matrix and  $\mathbf{B}$  is a  $p \times m$  column-orthogonal matrix*<sup>1</sup>.

Right multiplying by  $\mathbf{B}$  we get  $\mathbf{X}\mathbf{B} = \mathbf{H}$ . Thus the matrix  $\mathbf{B}$  assigns  $m$  different labels to each point. Before defining  $\mathbf{H}$  and  $\mathbf{B}$  in our setting, we want to give another definition of orthogonal clustering which is more aligned to how it is used in practical applications and is based on a tree like structure (though a very flexible one).

Consider the output from two clustering functions  $c_1$  and  $c_2$ , each of which partitions a set observations,

$$\begin{aligned} \mathbf{X} \xrightarrow{c_1} \mathbb{C}_1 &= \{b_{11}, b_{12}, b_{13}, \dots\} \\ \mathbf{X} \xrightarrow{c_2} \mathbb{C}_2 &= \{b_{21}, b_{22}, b_{23}, \dots\} \end{aligned}$$

where  $b_{ij}$  is the  $j$ th cluster from the  $i$ th clustering function. Now consider running  $c_1$  and then running  $c_2$  on each output cluster  $b_{1j}$ , an operation which we denote by  $c_1 \otimes c_2$ . The output is the cartesian products of the respective output sets,

$$\mathbf{X} \xrightarrow{c_1 \otimes c_2} \mathbb{C}_1 \times \mathbb{C}_2 = \{b_{11}b_{21}, b_{11}b_{22}, b_{11}b_{23}, b_{12}b_{21}, \dots\} \quad (4)$$

In fact, we could run  $m$  such clustering algorithms to obtain,

$$\mathbf{X} \xrightarrow{c_1 \otimes c_2 \otimes \dots \otimes c_m} \prod_i \mathbb{C}_i = \{C_1, \dots, C_K\}$$

where  $C_1 = \{b_{11}, b_{21}, \dots\}$  and so on. The  $C_i$ 's are clusters from the cartesian product, the assignment is not crucial (as long as there is one). Here  $K = \prod_i \text{card}(\mathbb{C}_i)$ .

<sup>1</sup>A non-square matrix  $\mathbf{U}$  is column-orthogonal if  $\mathbf{U}^T\mathbf{U} = \mathbf{I}$  and this is equivalent to the condition that each column of  $\mathbf{U}$  is orthogonal to every other column of  $\mathbf{U}$  but has a scalar product of 1 with itself.

**Definition 2** *Clusters  $\mathbb{C}_1$  and  $\mathbb{C}_2$  are orthogonal clusters and  $c_1$  and  $c_2$  are orthogonal clustering functions if  $\mathbb{C}_1 \times \mathbb{C}_2 = \mathbb{C}_2 \times \mathbb{C}_1$ . In other words, if the clusters produced by  $c_1$  followed by  $c_2$  are the same as  $c_2$  followed by  $c_1$ .*

Note that in general in the context of equation 4 we will not have  $b_{11}b_{21} = b_{21}b_{11}$ , except for the case where the features on which the clustering is being done are mutually exclusive. These clusterings are then orthogonal.

Following Gupta et al. (2007) we consider the clusters to be at the corners of a  $K$ -dimensional simplex. Projection of a simplex to a lower dimension is a *Simplicial Complex*.<sup>2</sup> Such constructs are commonly used in computer vision and CAD. Since orthogonal clustering aims to cluster in  $m$ -orthogonal directions, we consider a simplicial complex in  $\mathcal{R}^m$ . Note that this complex in  $\mathcal{R}^m$  has  $K$  nodes.

When clustering is done using CPCM, the clustering is associated with a distance metric which is given by the transformation  $\mathbf{A}$ . We can rewrite this  $\mathbf{A}$  as  $\mathbf{A} = \beta\beta^T$ , where  $\beta$  is the direction associated with the regression (refer section 2). Application of a CPCM step thus has a direction  $\beta$  corresponding to it. Consider two steps of CPCM,  $c_1$  and  $c_2$  with  $k_1$  and  $k_2$  splits and  $\beta_{c_1}$  and  $\beta_{c_2}$  as the corresponding directions respectively. Here  $K = k_1k_2$ ,  $m = 2$  and for  $c_1 \otimes c_2$  we have the direction  $\beta_{c_1 \otimes c_2}$ . Then on the lines of equation 2, we have a linear model given by

$$\mathbf{Z} = \mathbf{X}\beta_{c_1 \otimes c_2} + \text{noise} \quad (5)$$

where  $\mathbf{Z}_{.j}$  is the indicator for the membership to  $C_j$ . This in long hand really is,

$$\mathbf{Z}_{b_1} = \mathbf{X}\beta_{c_1} + \text{noise} \quad (6)$$

and for the  $k$ th split of  $c_1$

$$\mathbf{Z}_{b_{1k}} = \mathbf{X}_{b_{1k}}\beta_{c_2} + \text{noise} \quad (7)$$

where  $(\mathbf{Z}_{b_{11}}, \dots, \mathbf{Z}_{b_{1k_1}}) = \mathbf{Z}$ .  $c_1$  and  $c_2$  are orthogonal if the clusters from reversing the above operation are the same i.e,  $(\mathbf{Z}_{b_{11}}, \dots, \mathbf{Z}_{b_{1k_1}}) = (\mathbf{Z}_{b_{21}}, \dots, \mathbf{Z}_{b_{2k_2}})$ . In case of CPCM, when  $\beta_{c_1}$  and  $\beta_{c_2}$  are orthogonal then the steps  $c_1$  and  $c_2$  are orthogonal because the columns of  $\mathbf{Z}$  can be rearranged in this case. This leads us to an algorithm for orthogonal clustering based on CPCM.

<sup>2</sup>A simplicial complex  $\mathcal{K}$  in  $\mathbb{R}^n$  is a collection of simplices in  $\mathbb{R}^n$  such that (1) Every face of a simplex of  $\mathcal{K}$  is in  $\mathcal{K}$ , and (2) The intersection of any two simplices of  $\mathcal{K}$  is a face of each of them.

The algorithm solves an optimization problem which optimizes the blur ratio in the  $d$ th step. Instead of the writing as a ratio, we re-write blur ratio as a constrained convex program. At the  $d$ th step, we want to optimize the following program

$$\min_{\mathbf{A}_d, c} \sum_{k=1}^K \sum_{i:c(i)=k} (\mathbf{X}_i - \boldsymbol{\mu}_k) \mathbf{A}_d (\mathbf{X}_i - \boldsymbol{\mu}_k)^T$$

such that,

$$\begin{aligned} \sum_{i=1}^N (\mathbf{X}_i - \boldsymbol{\mu}) \mathbf{A}_d (\mathbf{X}_i - \boldsymbol{\mu})^T &= 1 \\ \mathbf{A}_d^T \mathbf{A}_{d-1} &= 0 \\ &\vdots \\ \mathbf{A}_d^T \mathbf{A}_1 &= 0 \end{aligned}$$

where  $\mathbf{A}_1, \dots, \mathbf{A}_{d-1}$  are the transformations found in the previous iterations. Running this from  $d = 1 \dots m$  we get our orthogonal clusters. This program also suggests that after a CPCM step, we can project out the  $\mathbf{X}$  in the CPCM direction and implement the next CPCM step on the orthogonal projection of  $\mathbf{X}$ .

The projection using  $\beta$  can be written as

$$\mathbf{P}_\beta = \beta(\beta^T \beta)^{-1} \beta^T \left( = \frac{\beta \beta^T}{\|\beta\|^2} \text{ for 1-D vector} \right) \quad (8)$$

Now, we can write  $\mathbf{X}$  as

$$\mathbf{X}^T = \mathbf{P}_\beta \mathbf{X}^T + (\mathbf{I} - \mathbf{P}_\beta) \mathbf{X}^T$$

Hence for the next iteration, we run the CPCM on  $\tilde{\mathbf{X}} = \mathbf{X}(\mathbf{I} - \mathbf{P}_\beta)^T$ . The Ortho-CPCM algorithm is given below.

---

**Algorithm 2** Ortho-CPCM
 

---

**Input:** Data  $\mathbf{X}$ . Set  $d = 1$  and  $\mathbf{X}_1 = \mathbf{X}$ .  
**repeat**  
     Implement CPCM on  $\mathbf{X}_d$  and get the direction  $\beta_d$ .  
     Compute the projection of  $\beta_d$ ,  $\mathbf{P}_{\beta_d}$  using equation 8.  
     Update  $\mathbf{X}_{d+1} = \mathbf{X}_d(\mathbf{I} - \mathbf{P}_{\beta_d})^T$ .  
     Increment  $d$ .  
**until**  $d = m + 1$

---

Ortho-CPCM at each step finds the best projection to a line which is orthogonal to previous cluster directions. Ortho-CPCM gives orthogonal clustering as stated by definition 2. We now show that this is also satisfies definition 1. As defined

above  $\mathbf{Z}_j$  is an indicator of cluster membership to  $C_j$ , and is an  $N$ -dimensional column vector. We modify this to  $\hat{\mathbf{Z}}_j^{c_i}$  to denote the predicted memberships from clustering algorithm  $c_i$ . We define  $\mathbf{H} = (\hat{\mathbf{Z}}_j^{c_1} \hat{\mathbf{Z}}_j^{c_2} \dots \hat{\mathbf{Z}}_j^{c_m})$  as an  $N \times m$  matrix. Corresponding to  $\hat{\mathbf{Z}}_j^{c_i}$  we have a column vector  $\beta_{c_i}(\cdot, j)$ . We now define  $\mathbf{B} = (\beta_{c_1}(\cdot, 1) \beta_{c_1}(\cdot, 2) \dots \beta_{c_m}(\cdot, m))$ . Since in Ortho-CPCM, the vectors  $\beta_{c_i}(\cdot, j)$  are orthogonal,  $\mathbf{B}$  is column-orthogonal. Also by using the linear model for CPCM, we get  $\mathbf{X}\mathbf{B} = \mathbf{H}$ , and hence definition 1 is also satisfied.

Figure 2 shows an example of a run of the algorithm. This example has 2 facet directions, one has 3 split and the other 2. In step 1 the Orth-CPCM finds the horizontal direction and splits the data accordingly. In step 2, the algorithm finds the vertical splits. Thus after the 2 steps all six clusters have been identified.

## 4. Related Work

An orthogonal clustering problem was solved by Zhang and Dong (2004). Extending the idea of Latent Semantic indexing, Zhang and Dong (2004) use a term-document matrix. They define cluster density as the Euclidean length of the dot-product of the cluster vector and the term-document matrix (Drineas et al., 1999). Orthogonal clusters are then defined as vectors which maximize this density, subject to them being orthogonal to each other. The solution to the problem turns out to be from the SVD of the term-document matrix. A related procedure is non-redundant clustering which also aims to find classes orthogonal to existing known classes. They do this by conditioning the clustering algorithm on the output of previous stages. Non-redundant clustering uses ideas based on conditional information bottleneck (Gondek & Hofmann, 2004) and conditional ensembles (Gondek & Hofmann, 2005). Derthick (1990) solves orthogonal clustering using the MDL principle but the facets are pre-determined. Vasilescu and Terzopoulos (2005) formulate multilinear ICA to solve the faceting problem, but here also the facets are given.

## References

- Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. In *Advances in neural information processing systems 15*, 585–592. Cambridge, MA: MIT Press.
- Derthick, M. (1990). *The minimum description length principle applied to feature learning and*

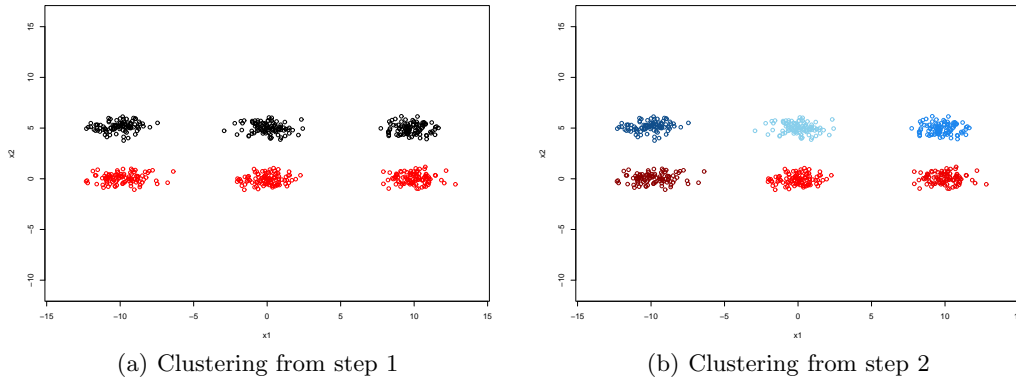


Figure 2. An example run is shown from (a)-(b). Data is a pattern of two by three Gaussian clouds. (a) Shows the clustering produced in step 1 from the Ortho-CPCM algorithm. Finds the correct clusters in the horizontal direction (2 in number) (b) Shows the clustering produced in step 1 from the Ortho-CPCM algorithm. Finds the correct clusters in the vertical direction (3 in number). The total number of clusters are 6 in this example.

*analogical mapping* (Technical Report ACT-AI-234-90). MCC.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., & Vinay, V. (1999). Clustering in large graphs and matrices. *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 291–299). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

Gondek, D., & Hofmann, T. (2004). Non-redundant data clustering. *ICDM* (pp. 75–82).

Gondek, D., & Hofmann, T. (2005). Non-redundant clustering with conditional ensembles. *KDD* (pp. 70–77).

Gupta, A., Foster, D. P., & Ungar, L. H. (2007). Unsupervised distance metric learning using predictability. *submitted to ICML '07, presented as poster in NESCAI '06*.

Hearst, M. A. (2006). Clustering versus faceted categories for information exploration. *Communications of the ACM*, 49.

Peng, J., & Xia, Y. (2005). A new theoretical framework for k-means-type clustering. In *Foundations and advances in data mining*, 79–96. Springer-Verlag.

Prieto-Diaz, R. (1990). Implementing faceted classification for software reuse. *Proceedings of the 12th International Conference on Software Engineering*.

Vasilescu, M. A. O., & Terzopoulos, D. (2005). Multilinear independent components analysis. *Proc.*

*Computer Vision and Pattern Recognition Conf. (CVPR '05)*.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems 15*, 505–512. Cambridge, MA: MIT Press.

Yee, K.-P., Swearingen, K., Li, K., & Hearst, M. (2003). Faceted metadata for image search and browsing. *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 401–408). New York, NY, USA: ACM Press.

Zhang, D., & Dong, Y. (2004). Semantic, hierarchical, online clustering of web search results. In *Proceedings of the 6th Asia Pacific Web Conference (APWEB)*. Hangzhou, China.