Viral Clustering: A Robust Method to Extract Structures in Heterogeneous Datasets

Vahan Petrosyan and **Alexandre Proutiere**
Royal Institute of Technology (KTH)
{vahanp, alepro}@kth.se

Abstract

Cluster validation constitutes one of the most challenging problems in unsupervised cluster analysis. For example, identifying the true number of clusters present in a dataset has been investigated for decades, and is still puzzling researchers today. The difficulty stems from the high variety of the dataset characteristics. Some datasets exhibit a strong structure with a few well-separated and normally distributed clusters, but most often real-world datasets contain possibly many overlapping non-gaussian clusters with heterogeneous variances and shapes. This calls for the design of robust clustering algorithms that could adapt to the structure of the data and in particular accurately guess the true number of clusters. They have recently been interesting attempts to design such algorithms, e.g. based on involved non-parametric statistical inference techniques. In this paper, we develop Viral Clustering (VC), a simple algorithm that jointly estimates the number of clusters and outputs clusters. The VC algorithm relies on two antagonist and interacting components. The first component tends to regroup neighbouring samples together, while the second component tends to spread samples in various clusters. This spreading component is performed using an analogy with the way virus spread over networks. We present extensive numerical experiments illustrating the robustness of the VC algorithm, and its superiority compared to existing algorithms.

Introduction

Clustering is an important technique in data mining and has many applications in machine learning, pattern recognition, bioinformatics, etc. For a given dataset $\mathbf{X}^{n\times p} = \{\mathbf{x}_i\}_{i=1}^n$ consisting of n observations of p variables, the main task of clustering is to divide the data into subgroups such that 'similar' observations appear in the same subgroup. In most clustering algorithms, the number of subgroups (clusters) should be specified in advance. In many cases however, the true number of clusters, denoted by k^* throughout the paper, has to be guessed. In turn, determining k^* is important in cluster analysis, and therefore, has been investigated extensively over the last few decades.

In this paper, we propose a novel algorithm, referred to as VC (Viral Clustering), that jointly addresses the problem of finding the true number of clusters, as well as identifying the clusters. The VC algorithm relies on two antagonist and interacting components. In the first component, Spread Virus, the clusters spread as a virus in a network, which favours the formation of large clusters, and in turn tends to remove clusters of small sizes. The second component, Suppress Virus, acts as k-mean algorithm, and tends to stop the epidemic spreading of clusters of the first component. By design, the VC algorithm exhibits a very high robustness, and performs well in datasets with possibly many overlapping non-gaussian clusters with heterogeneous variances and shapes. We present extensive numerical experiments illustrating the robustness of the VC algorithm, and its superiority compared to existing algorithms. VC accurately estimates the true number of clusters, and outputs clusters close to the true clusters.

Related Work

There is a large number of algorithms estimating the true number of clusters k^*. Most of them consist in running several instances of an arbitrary algorithm (usually k-means), with various values of k. The output (i.e., the clusters) of each instance is then processed and associated with a *validation* index that quantifies the quality of the output. k^* is finally obtained by maximizing/minimizing the index over k. Some researchers have done extensive comparisons of the various proposed validation indices. One of the earliest studies was carried out by Milligan and Cooper (1985) who provided an experimental summary of 30 different algorithms. All experiments were done on artificial datasets and the results showed that the method based on the index proposed in Calinski and Harabasz (1974) works best for determining k^*.

Bezdek et al. (1997) compared 23 different cluster validation indices. They used twelve datasets consisting of either three or six Gaussian clusters. Summarising the outcomes by counting the number of correct predictions of true k^*, they found that Bozdogan's (1993) AIC3 index performs the best. Hardy (1996) tested seven indices using six algorithms and six datasets with different structures. They discussed pros and cons of each method and recommended to use a combination of several cluster analysis techniques and validation

indices for determining the true number of clusters.

More recently, Arbelaitz et al. (2013) compared 30 different validation indices. The experiments in that paper were performed on both artificial and real-world datasets. Some of the indices in Arbelaitz et al. (2013) had already been discussed in Milligan and Cooper (1985). In particular, the Calinski and Harabasz (1974) index in Milligan and Cooper performed the third best on artificially generated datasets. It was only outperformed by Davis-Bouldin index (Davies and Bouldin 1979) and Silhouette index (Rousseeuw 1987). Various other papers compare up to ten indices with their own index (Tibshirani, Walther, and Hastie 2000; Sugar and James 2003; Kalogeratos and Likas 2012).

Next we describe the intuition behind some of the well-known and most successful validation indices. Calinski and Harabasz (1974) proposed the following index:

$$CH_k = \frac{B_k/(k-1)}{W_k/(n-k)},$$

where B_k and W_k denote the inter and intra-cluster sum of (distance) squares for k clusters. A large index here indicates that the dataset is well clustered, and therefore, k^* is determined by maximizing CH_k over k.

Hartigan (1975) proposed the following index:

$$HR_k = \left(\frac{W_k}{W_{k+1}} - 1\right)(n-k-1).$$

When the data has k^* distinct clusters, we expect a small drop from W_{k^*} to W_{k^*+1}, and hence, a small value of HR_k. As a crude rule of thumb, Hartigan suggested to determine k^* by finding the minimum number k such that $HR_k \leq 10$.

Rousseeuw (1987) proposed the Silhouette index:

$$SL_k = \frac{1}{n}\sum_{j=1}^{n}\frac{b_j - a_j}{max\{a_j, b_j\}},$$

where for the j-th observation, a_j is the average distance to other observations within the same cluster. Similarly, b_j is the average distance to the observations in the cluster the closest from the j-th observation (except for its own cluster). In case of well separated data, the difference $b_j - a_j$ gets closer to $max\{a_j, b_j\}$. Therefore, Rousseeuw suggested to choose k^* as the maximizer of the Silhouette index.

Krzanowski and Lai (1988) introduced the KL index defined as:

$$KL_k = \left|\frac{DIFF_k}{DIFF_{k+1}}\right|,$$

where $DIFF_k = (k-1)^{2/p}W_{k-1} - k^{2/p}W_k$. For a dataset with k^* clusters, we expect to see a big drop between W_{k^*-1} and W_{k^*} and a small drop between W_{k^*} and W_{k^*+1}. Thus, one can find k^* by maximizing the KL index.

Tibshirani, Walther, and Hastie (2000) proposed the gap index defined by:

$$GAP_k = B^{-1}\sum_{b=1}^{B} log(W_k^b) - log(W_k),$$

where each of W_k^b is the within cluster sum of squares generated from the reference data. As an example, consider the reference data has n uniformly distributed points in p dimensions, with k centers. It can be shown that $E[\log(W_k)] = log(pn/12) - (2/p)log(k) + C$ when the k centers are equally spaced. When the data have k separated clusters, the W_k will have a bigger drop from W_{k-1} in case of clustered data compared to reference data. Therefore, one will expect to see a higher value for GAP_k. However, instead of maximizing GAP index, Tibshirani, Walther, and Hastie suggested to choose k^* by minimizing k such that $GAP_k \geq GAP_{k+1} - s_{k+1}$, where $s_k = \sqrt{(1+1/B)sd(W_k^{1:B})}$.

A popular parametric method was proposed by Fraley and Raftery (2002). It finds k^* by fitting Gaussian Mixtures with EM algorithm and maximizing the BIC index for different values of k. The BIC index can be calculated as:

$$BIC_k = -2\log(\hat{L}_k) + q \cdot \log(n),$$

where $\hat{L}_k$ is the maximized value of the likelihood function and q is the number of independent parameters for the estimated model.

An information theoretical approach (the jump index) was proposed by Sugar and James (2003). This method is based on distortion rate, a measure of intra-cluster dispersion. The jump index can be calculated by

$$JUMP_k = d_k^{-y} - d_{k-1}^{-y}$$

where $d_k = p^{-1}\min_{c_1\cdots c_k} E[(X-c_x)^T\Sigma^{-1}(X-c_x)]$ is the average Mahalanobis distance per dimension between X and c_x; the c_i's are the cluster centers and y is a tuning parameter (usually $y = p/2$). In this method, k^* is estimated by maximizing the jump index over k. The authors provided some theoretical justifications of their method when the data consist of well-separated Gaussian clusters.

In this paper, we will compare our method with the aforementioned methods. Most of these methods predict k^* well under specific assumptions on the data. As it turns out, the proposed Viral Clustering algorithm outperforms existing methods and predicts the true number of clusters accurately without making any assumption on the underlying dataset.

Algorithm

In this section, we present Viral Clustering (VC), an algorithm that jointly estimates the number of clusters and outputs clusters. VC relies on two interacting components: (i) Spread Virus, a step where observations are attached to existing neighbouring clusters in an epidemic fashion; (ii) Suppress Virus, a step tending to regroup observations together as in the traditional k-means algorithm. VC alternates between these two steps until the cluster assignment remains roughly unchanged. Next we describe Spread Virus and Suppress Virus, and then present the main algorithm. In the algorithm, the current cluster assignments are described by $\mathbf{y} \in \{1, 2, \cdots, n\}^n$ where y_i is the identity of the cluster of sample $\mathbf{x}_i$. This identity can take any value from 1 to n (the total number of samples) since there are at most n clusters. Initially, VC starts with n clusters (one cluster per observation), i.e., $\mathbf{y} = \{1, 2, \cdots, n\}$.

Spread Virus. This step proceeds as follows. We sequentially consider each observation, and potentially modify its assignment to clusters. We denote by $\mathbf{s}$ the set of observations that have not been considered yet. To select an observation from $\mathbf{s}$, we choose uniformly at random (in $\mathbf{s}$) an observation belonging to the smallest cluster in the current cluster assignment $\mathbf{y}$. Let $l = \text{rand\_smallest}(\mathbf{y}, \mathbf{s})$ denote the index of this observation. We then select one of the m nearest neighbours of $\mathbf{x}_l$, say $\mathbf{x}_{l'}$, and assign $\mathbf{x}_l$ to the cluster $y_{l'}$, i.e., y_l becomes $y_{l'}$. The pseudo-code of the Spread Virus step is presented below. $\mathbf{D} \in \mathbb{R}^{n \times n}$ denotes the symmetric matrix such that $\mathbf{D}_{lk} = d(\mathbf{x}_l, \mathbf{x}_k)$ is the distance between observations $\mathbf{x}_l$ and $\mathbf{x}_k$; $\mathbf{D}_l$ is the l-th line of $\mathbf{D}$, and the function $\text{rand\_}m(\mathbf{D}_l)$ returns the index of one of the m nearest neighbours of $\mathbf{x}_l$ chosen uniformly at random.

Algorithm 1: Spread Virus

Input: $\mathbf{y} \in \{1, 2, \cdots, n\}^n, m$;
Initialization: $\mathbf{s} = \{1, 2, \cdots, n\}$;
while $\mathbf{s} \neq \emptyset$ **do**
  $l \leftarrow \text{rand\_smallest}(\mathbf{y}, \mathbf{s})$;
  $\mathbf{s} \leftarrow \mathbf{s} \backslash \{l\}$;
  $l' \leftarrow \text{rand\_}m(\mathbf{D}_l)$;
  $y_l \leftarrow y_{l'}$;
end
Output: $\mathbf{y}$;

Suppress Virus. This step simply consists in running one iteration of the k-mean algorithm starting from the current assignment $\mathbf{y}$. If there are k clusters in $\mathbf{y}$, we first compute the k centers of the clusters (the center of a cluster is the empirical average of the observations assigned to this cluster), and then assign each observation to the cluster with the closest center. The pseudo-code of this step is presented below. There, the function $\text{nb\_clusters}(\mathbf{y})$ returns the number k of clusters in $\mathbf{y}$, the function $\text{re\_index}(\mathbf{y})$ just redefines (arbitrarily) the indices of the clusters such that for all $l \in \{1, \ldots, n\}$, $y_l \in \{1, \ldots, k\}$, and finally the function $\text{centers}(\mathbf{y})$ returns the centers $\mu_1, \ldots, \mu_k$ of the k clusters.

Algorithm 2: Suppress Virus

Inputs: $\mathbf{y} \in \{1, 2, \cdots, n\}^n$;
$k \leftarrow \text{nb\_clusters}(\mathbf{y})$, $\mathbf{y} \leftarrow \text{re\_index}(\mathbf{y})$;
$(\mu_1, \ldots, \mu_k) \leftarrow \text{centers}(\mathbf{y})$;
for $l = 1 : n$ **do**
  $y_l \leftarrow \arg\min_{j \in \{1, \ldots, k\}} d(\mathbf{x}_l, \mu_j)$;
end
Output: $\mathbf{y}$;

Main Algorithm. The VC algorithm combines the Spread and Suppress Virus steps. Observe that the Spread Virus step tends regroup overlapping clusters into a single cluster. Indeed, in the area where two clusters overlap, Spread Virus mimics the competition between two epidemic processes, one for each cluster. Since we start modifying the assignment of observations in the smallest cluster, we give an advantage to large clusters in this competition. In turn, in the Spread Virus step, the number of clusters decreases. On the contrary, the Suppress Virus tends to freeze the various clusters around their centers, and keeps the number of clusters unchanged. The Suppress Virus step suppresses the *infected* observations around the cluster centers (by infected, we mean assigned to a cluster whose center is not the closest). The roles of the two steps are illustrated in Figure 1.

Algorithm 3: Viral Clustering

Input: $\mathbf{X} \in \mathbb{R}^{n \times p}, l_{spr}$;
Initialization: $i = 0, v = l_{spr}, \gamma_0 = 1, t_0 = n$,
$\mathbf{y}^0 = \{1, 2, \cdots, n\}, u = 30, m = \lfloor \log_2 n \rfloor, \zeta = 1.2$;
while $\gamma_i > 10^{-6}$ **do**
  $k_i \leftarrow \text{nb\_clusters}(\mathbf{y}^i)$;
  // spread or suppress virus
  if $v > 0$ **then**
    $\mathbf{y}^{i+1} \leftarrow$ Spread_Virus$(\mathbf{y}^i, m)$;
    $v \leftarrow v - 1$;
  else
    $\mathbf{y}^{i+1} \leftarrow$ Suppress_Virus$(\mathbf{y}^i)$;
    $v \leftarrow l_{spr}$;
  end
  $\Delta \leftarrow \#\{l : \mathrm{y}_l^{i+1} \neq \mathrm{y}_l^i\}/n$;
  // update t
  if $((i(mod)s = 1)$ & $(\gamma_{i-u} < \gamma_i))$ **then**
    $t_{i+1} \leftarrow t_i/\zeta$;
  else
    $t_{i+1} \leftarrow t_i$;
  end
  // update γ
  if $\Delta > k_i/t_{i+1}$ **then**
    $\gamma_{i+1} \leftarrow \gamma_i(1 + \Delta)$
  else
    $\gamma_{i+1} \leftarrow \gamma_i/2$;
  end
  $i \leftarrow i + 1$;
end
// suppress until convergence to k-means
while $\mathbf{y}^{i-1} \neq \mathbf{y}^i$ **do**
  $\mathbf{y}^{i+1} \leftarrow$ Suppress_Virus$(\mathbf{y}^i)$;
  $i \leftarrow i + 1$;
end
Output: $\mathbf{y}^i$

The proposed VC algorithm alternates between the two steps, until we do not see significant changes in the cluster assignment. More precisely, VC proceeds in rounds, and in each round, l_{spr} Spread Virus steps are made and one Suppress Virus step. For each point, we select $m = \lfloor \log_2 n \rfloor$ nearest neighbours in order to make the m-nearest neighbours graph connected within each cluster (Brito et al. 1997). The stopping rule is defined through a variable γ, updated after each step. The algorithm stops when γ becomes small enough (here 10^{-6}). After each step, γ is halved if the proportion Δ of observations that changed clusters is below a threshold, and is multiplied by $(1 + \Delta)$ otherwise. The threshold depends on the current number of clusters and a second variable t whose dynamics make sure that γ ulti-

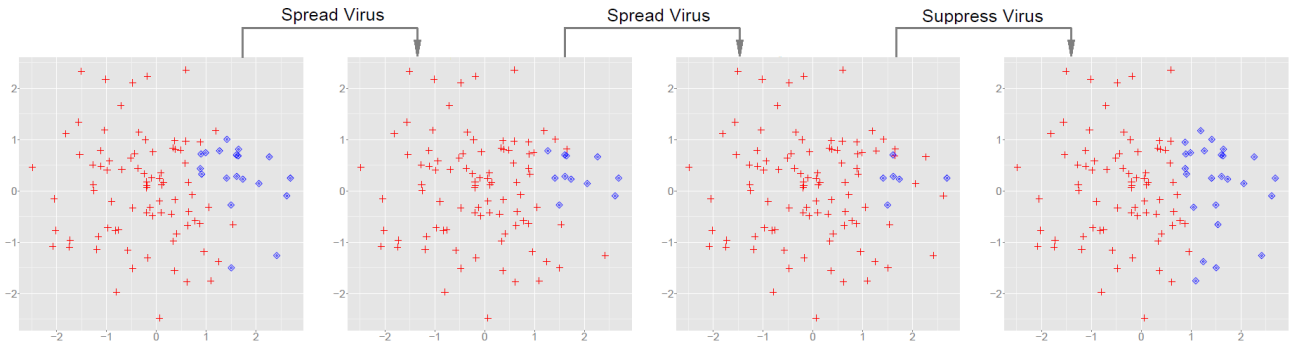

Figure 1: Cluster assignment dynamics under the VC algorithms over three consecutive steps (clusters are defined by the color or shape). In the first two Spread Virus steps, we tend to eliminate the smallest cluster. The Spread Virus step infects the neighbour points randomly, the Suppress Virus step removes the infected observations (those that are far from their cluster's center). Observations: Sampled from a standard bivariate Gaussian distribution, 100 observations.

mately decreases. t is updated every u steps and when γ has increased, in which case, the threshold is increased. The dynamics of t depend on the value of the parameter $\zeta > 1$ (the multiplicative decrease factor). u and ζ have little impact on the outcome of the algorithm, but slightly influence its convergence time. An example of typical dynamics for γ, t, and k (the current number of clusters) is presented in Figure 2.

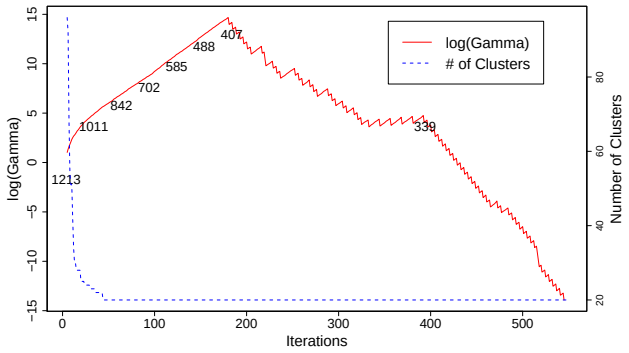

Figure 2: Evolution over time under the VC algorithm of γ ('Gamma' in the plot), t, and k. Observations: sampled from 20 t-distributed clusters. The values of t are the numbers presented near the red curve.

After γ becomes small enough, the VC algorithm finishes by running the Suppress Virus steps only, until the clusters are not changing anymore. This last step makes sure the algorithm to converge to one of the assignments obtained after convergence of the k-means algorithm. The pseudo-code of VC is provided in Algorithm 3.

Finally, we give some intuition for the VC algorithm. When the data is well-separated, Spread Virus is not capable to spread the virus from one cluster to another. Therefore, the virus spreads within the clusters only, and the algorithm does not suffer from the 'bad' initialization problem. When there is an overlap between clusters, the probability of spreading virus between clusters is much less than spreading within clusters. As a result, we can still predict k^* while avoiding the issue of 'bad' initialization.

Numerical Experiments

Datasets

A. For our experiments, we used ten artificially generated datasets from Tibshirani, Walther, and Hastie (2000) and Sugar and James (2003) (five datasets from each paper). Refer to these papers for a detailed description of the datasets.

B. We also created five other datasets with various properties as described in Table 1.

Dataset	Exp.	T	Beta	Mixture	Gaussian
Changing cluster size	✓	✓	✓	✓	✓
Changing cluster gaps	✓	✓	✓	✓	
Changing variances	✓		✓	✓	
Correlated variables					✓
Changing distributions				✓	
High dimensional					✓
Outliers	✓	✓		✓	

Table 1: Properties of five artificial datasets.

All five datasets have 20 clusters and in each cluster there are from 40 to 80 observations, drawn randomly according the different types of distributions, e.g., Gaussian, Exponential, Beta, Student's t distributions. A random realization of the first four datasets are presented in Figure 3. There, observations are 2D vectors, and the clusters are centered near $(\mathbf{c}_1, \mathbf{c}_2)$ where $\mathbf{c}_1 \in \{a_1, \dots, a_5\} = \{0, 3, 6, 9, 18\}$ and $\mathbf{c}_2 \in \{b_1, \dots, b_4\} = \{0, 3, 6, 15\}$. Now an observation $\mathbf{x} = (x_1, x_2)$ from cluster $j \in \{1, \dots 20\}$ is drawn from the following distributions:

1. First dataset (Exponential): Let λ be selected uniformly at random in $\{1, 3\}$ (we use the same λ for observations in the same cluster):

$$x_1 \sim Exp(\lambda) + 1 + a_{1+j(mod)5},$$
$$x_2 \sim Exp(\lambda) + 1 + b_{1+j(mod)4}.$$

2. Second dataset (T):

$$x_1 \sim 0.6 \cdot t(df = 3) + 1.5 + a_{1+j(mod)5},$$
$$x_2 \sim 0.6 \cdot t(df = 3) + 1.5 + b_{1+j(mod)4}.$$

3. Third dataset (Beta): Each parameter $\alpha_1, \alpha_2, \beta_1, \beta_2$ are chosen uniformly at random in $\{2, 3, 4, 5\}$,

$$x_1 \sim 3.75Beta(\alpha_1, \beta_1) + a_{1+j(mod)5},$$
$$x_2 \sim 3.75Beta(\alpha_2, \beta_2) + b_{1+j(mod)4}.$$

4. Fourth dataset (Mixture): for each cluster, we randomly choose the distribution among the 3 distributions described above and generate observations accordingly.

5. Fifth dataset (Gaussian): observations are drawn from 50-dimensional multivariate normal distributions with covariance matrix of 1 in the diagonal and 0.5 elsewhere; the cluster centers are at $\{3, 6, 9, \cdots, 60\}$ in each dimension.

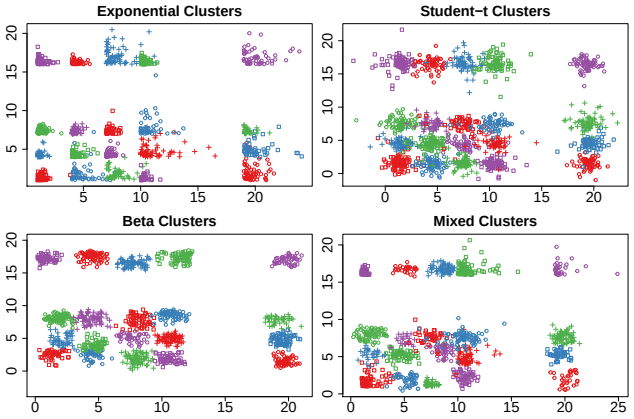

Figure 3: Random realization of the first four datasets. Clusters are defined by the color and shape combinations.

C. We finally tested our algorithm on five real world datasets, collected from UCI machine learning repository (Lichman 2013). These datasets include: iris ($n = 150, p = 4, k^* = 3$), breast cancer ($n = 683, p = 9, k^* = 2$), multiple features Furie coefficients ($n = 2000, p = 76, k^* = 10$), wine ($n = 178, p = 13, k^* = 3$), ruspini ($n = 75, p = 2, k^* = 4$). Refer to UCI machine learning repository for more information about the datasets.

Predicting the true number k^* of clusters

We evaluate here the accuracy of the index-based prediction methods mentioned in the related work section. These methods rely on a particular underlying clustering algorithm. In our experiments, we used the k-means algorithm, except for the BIC index where the EM algorithm (fitting mixture of Gaussians) was used.

A. We have done extensive experiments on the 10 datasets used in Tibshirani, Walther, and Hastie (2000) and Sugar and James (2003). We used 50 random realizations for each dataset and summarized our findings by calculating how many times the various algorithms (see the Related Work section) predicted k^* correctly, see Table 2. For the VC algorithm, we have chosen $l_{spr} = 1$ when $p = 10$ and $l_{spr} = 3$ when $p = 2$, or 3. Only in Gap5, we used $l_{spr} = 20$.

Data, k^*, p	KL	CH	HR	SL	Gap	BIC	Jump	VC
Gap1, 1, 10	0	0	0	0	50	50	0	50
Gap2, 3, 2	36	50	50	45	43	50	49	48
Gap3, 4, 3	22	28	29	20	19	50	50	49
Gap4, 4, 10	31	23	27	22	28	50	48	47
Gap5, 2, 3	50	0	0	50	50	50	0	50
Jump1, 5, 2	5	46	0	13	1	48	45	39
Jump2, 5, 10	25	0	2	0	34	50	42	47
Jump3, 4, 2	41	50	50	50	7	50	50	48
Jump4, 4, 2	9	50	50	21	25	46	49	47
Jump5, 4, 2	23	30	47	39	11	11	42	48

Table 2: Number of times k^* is correctly predicted (out of 50 experiments) by different validation indices. All datasets have between 100 and 200 observations.

BIC, Jump and VC perform well for these datasets. However, the BIC algorithm is based on Gaussian mixture models and works well when clusters are normally distributed and not high-dimensional. For example, it has poor performance in the Jump5 dataset where clusters have exponential distribution. The Jump algorithm tends to overestimate the number of clusters when a large prediction range is chosen (in these experiments we chose k^* from the prediction range one to ten). Note that in VC there is no need to specify the prediction range of k.

	Exp	T	Beta	Mixture	Gaussian
KL	4	1	0	0	1
CH	0	2	4	3	0
HR	0	0	0	0	0
SL	3	0	0	0	0
Gap	0	0	0	0	0
BIC	0	3	15	7	0
Jump	4	7	10	5	0
VC	50	47	33	32	50
	$\mu_k \pm \sigma_k$	$\mu_k \pm \sigma_k$	$\mu_k \pm \sigma_k$	$\mu_k \pm \sigma_k$	$\mu_k \pm \sigma_k$
KL	20.9±11	16±12.6	18.9±11.8	17.8±12.6	23.7±5.2
CH	31.5±5.3	25.5±4.5	24.9±3.8	26.2±6.3	24±2.5
HR	4.9±5.2	3.2±0.6	3.9±2.5	5.5±5.9	3.6±1.5
SL	19.2±7.4	3.6±2.3	5.7±4.5	9.9±5.8	2±0
Gap	2.1±1.5	1.6±0.9	1.9±1.1	1.9±1.2	2.4±1.7
BIC	23.3±1.5	22.4±5.7	21.3±2.3	20.6±2.6	1±0
Jump	26.4±5.6	21.6±1.9	20.3±4.2	23.7±6.6	39.2±0.9
VC	20±0	19.9±0.1	19.7±0.6	19.7±0.7	20±0

Table 3: **Top:** Number of times k^* is correctly predicted (out of 50 experiments) by different validation indices. **Bottom:** Mean and standard deviation of the predictions.

B. The experimental results for the five artificial datasets are shown in Table 3. Again, we randomly created 50 datasets and presented the number of correct predictions. In addition, we provide the mean and the standard deviation of these 50 predictions. Here, the performance of VC is outstanding. While all other algorithms perform really poorly, VC correctly predicted k^* most of the time, in all five cases.

Even when the VC prediction was wrong, it was actually very close to the true number of clusters, $k^* = 20$ (check the $\mu_k \pm \sigma_k$ in Table 3).

C. We have similar performance results for the real-world datasets. Table 4 shows the predicted number of clusters for different validation indices. VC did not correctly predict k^* only in multiple features Furie coefficients dataset, but its prediction was at least as good as its competitors. For all real world and five artificially generated datasets we used $l_{spr} = 3$.

Data	KL	CH	HR	SL	Gap	BIC	Jump	VC	k^*
Cancer	2	2	3	2	6	7	10	2	2
Iris	4	3	3	2	3	2	8	3	3
Furie	11	2	9	2	7	20	20	11	10
Ruspini	4	4	4	4	4	5	4	4	4
Wine	10	3	3	3	3	8	10	3	3

Table 4: Number of clusters predicted by validation indices for real-world datasets. The last column presents the true number of clusters.

Accuracy of various clustering algorithms

We compared the accuracy of VC with 4 well-known algorithms, namely, DP-means (Kulis and Jordan 2012), Gaussian Mixture Models (GMM) (Fraley and Raftery 2002), k-means with initialization based on Hartigan and Wong (1979), and Spectral Clustering (Ng, Jordan, and Weiss 2001). Except for Kulis and Jordan, these algorithms need to specify the number of clusters in advance. So, for the five artificial datasets, we run our experiments with parameter $k = 20$, the true number of clusters. For Kulis and Jordan, k is controlled by parameter λ and therefore, we run this algorithm with λ such that it provides 20 clusters.

We used two popular metrics to measure the accuracy of each clustering algorithms. The first method, Adjusted Rand (AR) Index (Hubert and Arabie 1985), is based on the *confusion* matrix that compares the clusters identified by the algorithm and the true clusters. The Variation of information (Vi), (Meilă 2007) is based on the information theoretical notions of entropy and mutual information.

We tested the accuracy of all five artificially generated datasets using both comparison metrics. For each dataset, we simulated 100 random realiztions and presented the boxplots of accuracies in Figure 4. All four competing algorithms suffer from bad initialization. In addition, GMM does not perform well due to non-gaussian clusters (first four datasets) and estimating well the 50-dimensional covariance matrix (fifth dataset). For all 5 datasets, VC outperforms the other clustering algorithms. In some cases, even the lower quantile of VC accuracy is still higher than the upper quantile of its competitiors.

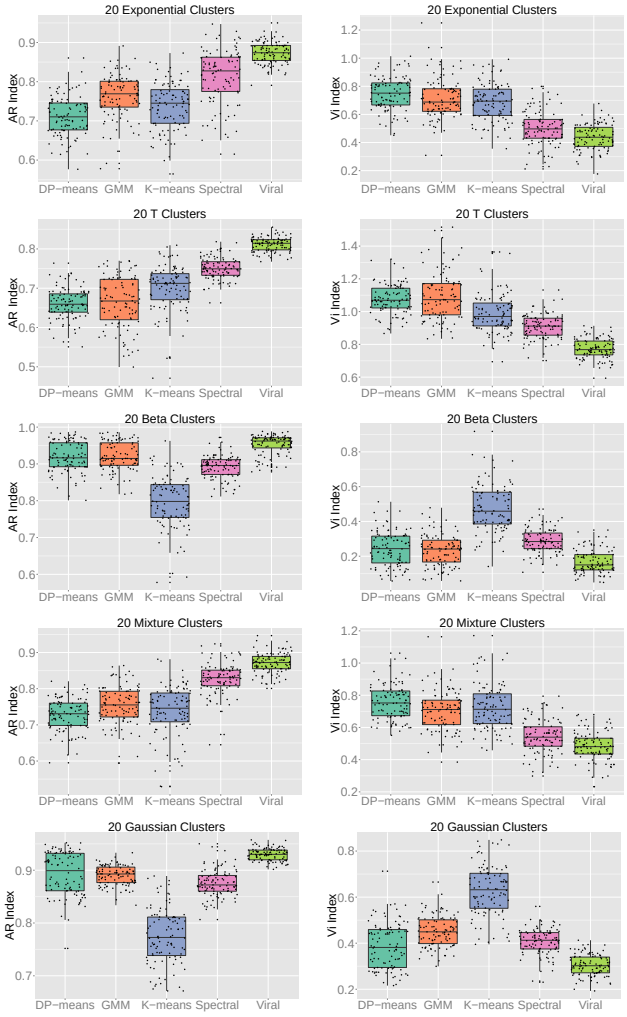

Figure 4: Accuracy of the various clustering algorithms for the artificial datasets (100 realizations for each dataset). **Left column**: Adjusted Rand Index (the higher, the better) and **right column**: Variation of Information Index (the lower, the better).

Conclusion

We proposed a novel algorithm, VC, that determines the number of clusters and performs clustering. Compared to other k-means based cluster validation methods, VC does not require multiple initializations. VC does not require to run k-means for different values of k. We have shown, through extensive numerical experiments, that VC is robust and outperforms existing algorithms both when estimating the true number of clusters, and when predicting clusters. In future work, we shall investigate how to automatically and optimally tune the main parameter l_{spr} of the VC algorithm, and also plan to derive theoretical guarantees for its performance. Finally, it is worth searching for ways to improve the stopping criterion of the algorithm, and hence to increase its convergence rate.

References

Arbelaitz, O.; Gurrutxaga, I.; Muguerza, J.; Pérez, J. M.; and Perona, I. 2013. An extensive comparative study of cluster validity indices. *Pattern Recognition* 46(1):243–256.

Bezdek, J. C.; Li, W. Q.; Attikiouzel, Y.; and Windham, M. 1997. A geometric approach to cluster validity for normal mixtures. *Soft Computing* 1(4):166–179.

Bozdogan, H. 1993. Choosing the number of component clusters in the mixture-model using a new informational complexity criterion of the inverse-fisher information matrix. In *Information and Classification*, Studies in Classification, Data Analysis and Knowledge Organization. Springer Berlin Heidelberg. 40–54.

Brito, M.; Chvez, E.; Quiroz, A.; and Yukich, J. 1997. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics and Probability Letters* 35(1):33 – 42.

Calinski, T., and Harabasz, J. 1974. A dendrite method for cluster analysis. *Communications in Statistics* 3(1):1–27.

Davies, D. L., and Bouldin, D. W. 1979. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1(2):224–227.

Fraley, C., and Raftery, A. E. 2002. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association* 97(458):611–631.

Hardy, A. 1996. On the number of clusters. *Computational Statistics and Data Analysis* 23(1):83–96.

Hartigan, J. A., and Wong, M. A. 1979. A k-means clustering algorithm. *Applied Statistics* 28(1):100–108.

Hartigan, J. A. 1975. *Clustering Algorithms*. New York, NY, USA: John Wiley & Sons, Inc.

Hubert, L., and Arabie, P. 1985. Comparing partitions. *Journal of Classification* 193–218.

Kalogeratos, A., and Likas, A. 2012. Dip-means: an incremental clustering method for estimating the number of clusters. *Advances in Neural Information Processing Systems (NIPS)* 2402–2410.

Krzanowski, W. J., and Lai, Y. T. 1988. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics* 44:23–34.

Kulis, B., and Jordan, M. I. 2012. Revisiting k-means: New algorithms via bayesian nonparametrics. *Proceedings of the 29th International Conference on Machine Learning (ICML)* 513–520.

Lichman, M. 2013. UCI machine learning repository.

Meilă, M. 2007. Comparing partitions. *Journal of Multivariate Analysis* 98(5):873–895.

Milligan, G., and Cooper, M. 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika* 50(2):159–179.

Ng, A. Y.; Jordan, M. I.; and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)* 849–856.

Rousseeuw, P. J. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20:53–65.

Sugar, C. A., and James, G. M. 2003. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association* 98(463):750–763.

Tibshirani, R.; Walther, G.; and Hastie, T. 2000. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society* 63(2):411–423.