Fast Approximate Text Document Clustering Using Compressive Sampling

Laurence A.F. Park

School of Computing and Mathematics,
University of Western Sydney, Australia
lapark@scm.uws.edu.au
http://www.scm.uws.edu.au/~lapark

Abstract. Document clustering involves repetitive scanning of a document set, therefore as the size of the set increases, the time required for the clustering task increases and may even become impossible due to computational constraints. Compressive sampling is a feature sampling technique that allows us to perfectly reconstruct a vector from a small number of samples, provided that the vector is sparse in some known domain. In this article, we apply the theory behind compressive sampling to the document clustering problem using k-means clustering. We provide a method of computing high accuracy clusters in a fraction of the time it would have taken by directly clustering the documents. This is performed by using the Discrete Fourier Transform and the Discrete Cosine Transform. We provide empirical results showing that compressive sampling provides a 14 times increase in speed with little reduction in accuracy on 7,095 documents, and we also provide a very accurate clustering of a 231,219 document set, providing 20 times increase in speed when compared to performing k-means clustering on the document set. This shows that compressive clustering is a very useful tool that can be used to quickly compute approximate clusters.

1 Introduction

Clustering computational complexity is dependent on the number of objects in the set and the number of features of each object. Therefore, as the number of features grows, the feasibility of applying a clustering algorithm reduces. To perform clustering, the data must be repeatedly scanned while the clusters are refined therefore it is also important that we have enough memory for the computation to avoid lengthy disk accesses.

As technology advances, the size of data to be processed also increases. Text document databases are growing at a rapid rate, therefore, it is crucial that we derive document clustering algorithms that are able to process and cluster these large data sets.

Compressive sampling [4,5,3,7,1] is a new concept in Information theory that states that we are able to perfectly reconstruct a vector from only a few samples, when using an appropriate incoherent sampling scheme.

In this article, we introduce compressive clustering; a method of clustering using incoherent samples of the features, that provides a close approximation to the clusters that would have been found on the unsampled data. We show that we are able to apply compressive clustering to very high dimensional spaces and obtain very accurate cluster estimates in a fraction of the time.

We make the following contributions:

- We provide a generalised algorithm for compressive clustering (Section 3),
- A radial k-means algorithm for complex vector spaces (Section 3.2), and
- Use of compressive clustering for document clustering using radial k-means clustering with discrete Fourier and discrete Cosine sampling (Section 3.3).

The article will proceed as follows: Section 2 provides a brief introduction to compressive sampling, section 3 introduces our compressive clustering algorithm and how it can be applied to document clustering using radial k-means. Section 4 provides experimental results showing the performance of compressive clustering for document clustering, and finally section 5 presents the use of compressive clustering on a large document set.

2 Coherence and Random Projections

Compressive sampling is a new sampling technique that has gained popularity in the image processing domain. It is an generalisation of the Nyquist sampling theorem that is not restricted to band-limited signals. In this section we will examine Nyquist's theorem and Compressive sampling.

2.1 Sampling Cyclic Signals

The Nyquist-Shannon sampling theory is at the base of Information Theory. The theory states that a signal containing no frequencies greater than b hertz, can be perfectly reconstructed by sampling at a rate of at least $2b$ hertz.

Figure 1 shows a simple example of a 2 hertz signal in the form of a sinusoidal wave. The Fourier transform of this signal would contain one non-zero frequency component, therefore when sampling the signal, we must ensure that we capture this single non-zero frequency value. If we can do this, and assume that the remaining frequency values are zero, then we are able to perfectly reconstruct the sinusoidal wave. The Nyquist-Shannon theorem tells us that we are able to capture this nonzero frequency component by sampling at at least two times the frequency, which in this case is a rate of 4 hertz.

2.2 Sampling Sparse Signals

The Nyquist-Shannon sampling theorem allows us to capture all frequency components up to half of our sampling rate. This allows us to efficiently encode low frequency signals, but what if there were only a few non-zero frequency components that were scattered across the frequency domain. By using the Nyquist-Shannon sampling theorem, we would have to sample at a high rate in order to

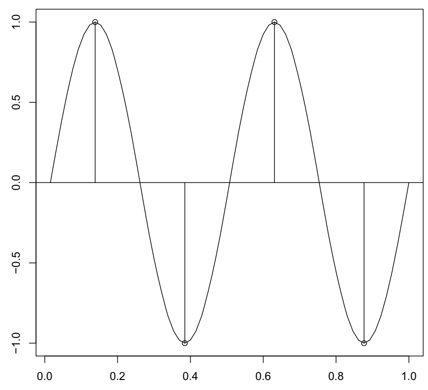

Fig. 1. A 2 hertz signal, being sampled at a rate of four hertz. According to the Nyquist-Shannon sampling theorem, we can perfectly reconstruct the signal using these four samples.

capture only a few non-zero frequency components. Or what if the frequency domain of the signal was dense (many non-zero components), but there was some other domain in which the signal has a sparse representation?

A recent advancement in Information Theory is that of Compressive Sampling [4,5,3,7,1]. Compressive sampling theory states that given a signal that is shown to be sparse in a known domain, we are able to take a small number of random samples of the signal and successfully reconstruct the signal to its original state. Stated more formally, we have:

$$\boldsymbol{x} = \min \|\boldsymbol{z}\|_1, \text{s.t. } \boldsymbol{\xi} = \Phi\Psi\boldsymbol{z} \tag{1}$$

where $\boldsymbol{y} = \Psi\boldsymbol{x}$ is the original signal and $\boldsymbol{x}$ is sparse, Φ is the sampling function, $\boldsymbol{\xi}$ is the set of samples, and $\|\cdot\|_1$ is the l_1 norm. To obtain perfect reconstruction, we must ensure that $\boldsymbol{x}$ is S-sparse (implying that $\boldsymbol{x}$ has at most S non-zero values), and that the transforms Φ and Ψ are incoherent.

Before we proceed lets examine the case where the Fourier transform of a signal is sparse, but the non-zero components are spread evenly across the whole spectrum. We have said that the Nyquist-Shannon sampling theorem would require us to sample at a high rate, due to some of the frequency components being of high frequency. By using Compressive Sampling, we know that the Fourier Transform of the signal is sparse, therefore we let Ψ be the Fourier transform and $\boldsymbol{x}$ be the transformed signal. We can construct a sampling function Φ by randomly sampling rows of the identity matrix. This implies that if we take K random samples of our signal $\Phi\boldsymbol{x}$, we are able perfectly reconstruct it given that K is large enough.

It has been shown that when using the Fourier transform as Ψ and a sampled identity matrix as Φ, we obtain the relationship for perfect reconstruction:

$$K \geq CS \log N$$

where K is the number of samples, N is the length of the original signal and C is a constant.

Therefore, given that $S = 10$, $N = 1000$, then the number of samples K required for perfect reconstruction is proportional to 70. Note that if not enough samples are taken, the resulting reconstructed sparse vector will be similar to a thresholded version of $\boldsymbol{x}$ due to the use of the l_1 norm in the minimisation.

For the example above, we chose a transform and sampling matrix that were maximally incoherent. Coherence is a measure of basis similarity that computes the smallest angle between the basis vectors from the two basis sets. The coherence of Φ and Ψ is given as:

$$\mu(\Phi, \Psi) = \sqrt{N} \max_{1 \leq i,j \leq N} |\langle \boldsymbol{\phi_i}, \boldsymbol{\psi_j} \rangle|$$

where $\boldsymbol{\phi_i}$ and $\boldsymbol{\psi_j}$ are basis vectors in the transformation Φ and Ψ, and $\mu(\Phi, \Psi) \in [1, \sqrt{N}]$. Therefore given the Fourier transform and identity matrix as Ψ and Φ, we obtain $\mu(\Phi, \Psi) = 1$, being maximally incoherent. For other choices of Ψ and Φ, where the coherence is greater than one, we have the generalised relationship:

$$K \geq C\mu^2(\Phi, \Psi) S \log N$$

Therefore, we can see that the choice of the transformation and sampling matrices are crucial in reducing the number of samples required for perfect reconstruction. The more incoherent the two basis sets are, the more they spread over each other. For example, the Fourier and identity basis are maximally incoherent since each Fourier coefficient is a combination of all identity basis vectors.

3 Compressive Clustering

Now that we have an understanding of compressive sampling, we will introduce the use of compressive sampling for clustering.

We have stated that clustering algorithms require repetitive access to the data while clustering, therefore as the size of the data grows, so to does the time and storage required to compute the clusters. This implies that there is a limit as to the size of data we can cluster, that is dependent on the current computation and storage available.

Compressive sampling has provided us with the concept of incoherent sampling. By using incoherent samples, we are able to capture enough information to perfectly reconstruct the vector from the samples, given knowledge of the sampling scheme. We can apply this theory to obtain a low dimensional feature space in which we can perform clustering. Using the small dimensional representation of the clusters, we are able to reconstruct the vectors in the original high dimensional space.

Our compressive clustering algorithm is as follows: Given a data set $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\}$, a transform Ψ, such that $\boldsymbol{y}_i = \Psi \boldsymbol{x}_i$ where each $\boldsymbol{x}_i$ are sparse, and a sampling function Φ that is incoherent to Ψ:

1. Sample the features of the vector space using the sampling function to obtain $\boldsymbol{\xi}_i$, where $\boldsymbol{\xi}_i = \Phi \boldsymbol{y}_i$.
2. Cluster the sampled space $\{\boldsymbol{\xi}_1 \ldots \boldsymbol{\xi}_M\}$.
3. For each vector $\boldsymbol{\xi}_i$ in cluster $\mathcal{C}_m$, reconstruct the original vector $\boldsymbol{x}_i$ using equation 1.
4. Compute each cluster definition in the unsampled space based on the cluster vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$.
5. Re-assign each unsampled vector based on the unsampled cluster definitions.

The cluster definition is what defines the cluster (e.g. a hyperplane, a point in the space, a direction in the space). We also make use of the cluster definition when assigning new vectors to a cluster. Note that step 2 can be performed using compressive sampling reconstruction, or by accessing the original data if possible.

Using this algorithm, we are able to perform the clustering in the reduced space, requiring less computation and less memory. The algorithm requires:

– one pass over the data to sample each vector which is passed to the clustering method.
– a second pass over the data to compute each cluster definition.
– a third pass over the data to assign each vector to a cluster based on the cluster definitions.

Therefore, rather than requiring many scans of high dimensional data set, we are able to cluster the data in a reduced space requiring only three scans of the high dimensional data.

3.1 Document Clustering

The common representation of documents as vectors uses a vector space where each dimension represents a unique term in the document collection. For example, a document existing in a document collection containing 200,000 unique terms, is represented using a 200,000 dimensional vector. We can see that when using this vector space, the dimension of the vector space is related to the number of documents in the collection, and can only grow as new documents are added.

Using this vector representation also leads to very sparse document vectors (containing many zero elements). If a document contains 100 unique terms, then its associated document vector in the 200,000 dimensional space would contain 199,900 zero elements. It is this feature that leads to high compression ratios when constructing a document index for information retrieval.

It is common for less that 1% of document vector elements to contain nonzero values. Therefore it is safe to assume that document vectors are a sparse representation of the documents.

Document vector elements contain the frequency (or weighted frequency) of the associated term in the associated document. Based on this, it does not make sense to use Euclidean k-means to cluster the document set, as Euclidean distance is a geometric distance between two points. The value of zero in a vector is simply a position of a point to the Euclidean distance metric. In our case, zero is not a position, but an absence of a term. Therefore, we provide a modification of the k-means algorithm using the angle between two vectors as a measure of similarity.

3.2 Complex Radial K-means

In this section, we present a form of k-means that uses the angle between vectors as a measure of similarity, and is able to cluster vectors existing in a complex vector space.

K-means is an iterative process that records a cluster centre vector $\boldsymbol{c}_m$ for each cluster and adjusts it until stability is reached.

The first part of any k-means process is the initialisation of the centre vectors. In this article, we take the approach of Hartigan and Wong [8], and adapt it for use with the vector angle similarity metric. Given the mean vector $\bar{\boldsymbol{x}} = \sum_i \boldsymbol{x}_i$ and the angle between each vector and the mean vector $\theta_i = \bar{\boldsymbol{x}}\angle\boldsymbol{x}_i$, the cluster centres are initialised using:

$$\boldsymbol{c}_m = \boldsymbol{x}_{\text{round}(1+(m-1)N/C)}$$

where the vectors $\boldsymbol{x}_i$ are sorted by θ_i (the angle θ_1 associated to $\boldsymbol{x}_1$ is the smallest and the angle θ_N associated to the vector $\boldsymbol{x}_N$ is the largest angle), and round(x) rounds fractional values to their nearest integer.

Using this initialisation, we can compute the radial k-means clusters using the iterative process:

$$\begin{aligned}\boldsymbol{c}_m &\leftarrow \sum_{i\in\mathcal{C}_m} \boldsymbol{x}_i \text{ for } m \in \{1,\ldots,C\}\\ \mathcal{C}_m &\leftarrow \{x_i | m = \underset{m}{\text{argmin}}\,(\boldsymbol{c}_m\angle\boldsymbol{x}_i)\} \text{ for } m \in \{1,\ldots,C\}\end{aligned}$$

where $\mathcal{C}_m$ is the set of vectors associated to cluster m.

We define the angle between any two complex vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as:

$$\cos\left(\boldsymbol{x}_i\angle\boldsymbol{x}_j\right) = \frac{\text{Re}(\langle\boldsymbol{x}_i,\boldsymbol{x}_j\rangle)}{\|\boldsymbol{x}_i\|_2\|\boldsymbol{x}_j\|_2}$$

where $\|\boldsymbol{x}_i\|_2 = \left(\sum_j |x_{i,j}|^2\right)^{1/2}$ and $\text{Re}(a+ib) = a$ for real values a and b.

3.3 Approximate k-means Document Clustering

By using radial k-means as our clustering algorithm, in conjunction with our compressive clustering algorithm, we obtain a compressive clustering algorithm suitable for documents:

1. Sample the features of the vector space using the sampling function $\varPhi$.
2. Perform k-means on the sampled space.
3. For each vector $\boldsymbol{\xi}_i$ in cluster $\mathcal{C}_m$, reconstruct the original vector $\boldsymbol{x}_i$.
4. Compute the cluster centre $\boldsymbol{c}_m = \sum_{i \in \mathcal{C}_m} \boldsymbol{x}_i$.
5. Compute the unsampled space clusters using the unsampled space cluster centres.

To complete the compressive clustering algorithm we must choose an appropriate transform and sampling matrices. The transform matrix $\varPsi$ provides the relationship between the sparse domain and the data domain. We have stated in section 3.1 that the document vectors are sparse, therefore a suitable transform matrix is the identity matrix ($\varPsi = I$).

The sampling matrix must be incoherent to the transform matrix, therefore the perfect choice is a random sample of the discrete Fourier basis:

$$\xi_{j,k} = \sum_{n=1}^{N} y_{j,n} e^{-i2\pi(k-1)(n-1)/N}$$

where $y_{j,n}$ is the nth element of vector $\boldsymbol{y}_j$, and $\xi_{j,k}$ is the kth Discrete Fourier Transform (DFT) sample of vector $\boldsymbol{y}_j$. Noting that the discrete Fourier basis is complex, we will also examine the discrete Cosine basis:

$$\xi_{j,k} = \sum_{n=1}^{N} y_{j,n} \cos\left(\pi(k-1)(2n-1)/2N\right)$$

where in this case $\xi_{j,k}$ is the kth Discrete Cosine Transform (DCT) sample of the vector $\boldsymbol{y}_j$.

Given a transformation matrix $\varPhi$ and vectors $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$, we can compute the transformed vectors: $\boldsymbol{\xi}_i = \varPhi \boldsymbol{y}_i$ and $\boldsymbol{\xi}_j = \varPhi \boldsymbol{y}_j$. If we define $\boldsymbol{\xi}_i \angle \boldsymbol{\xi}_j$ as the angle between vectors $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$, then:

$$\begin{aligned}
\cos\left(\boldsymbol{\xi}_i \angle \boldsymbol{\xi}_j\right) &= \frac{\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j \rangle}{\|\boldsymbol{\xi}_i\|_2 \|\boldsymbol{\xi}_j\|_2} \\
&= \frac{\langle \varPhi \boldsymbol{y}_i, \varPhi \boldsymbol{y}_j \rangle}{\|\varPhi \boldsymbol{y}_i\|_2 \|\varPhi \boldsymbol{y}_j\|_2} \\
&= \frac{\boldsymbol{y}_i^T \varPhi^T \varPhi \boldsymbol{y}_j}{\sqrt{\boldsymbol{y}_i^T \varPhi^T \varPhi \boldsymbol{y}_i} \sqrt{\boldsymbol{y}_j^T \varPhi^T \varPhi \boldsymbol{y}_j}}
\end{aligned}$$

From this expansion, we can clearly see that if $\varPhi$ is a unitary transformation, then $\cos\left(\boldsymbol{\xi}_i \angle \boldsymbol{\xi}_j\right) = \cos\left(\boldsymbol{y}_i \angle \boldsymbol{y}_j\right)$.

The Discrete Fourier transform and Discrete cosine transform are unitary transformations, implying that vector norms are preserved when the transformation is performed. By sampling the rows of the transformation, we obtain a matrix that is no longer unitary, but has been shown to be approximately orthogonal [5]:

$$(1-\delta)\|\boldsymbol{y}\|_2^2 \leq \|\varPhi \boldsymbol{y}\|_2^2 \leq (1+\delta)\|\boldsymbol{y}\|_2^2$$

for small values of δ.

Therefore, if Φ is approximately orthogonal, we can show that $\cos(\boldsymbol{\xi}_i \angle \boldsymbol{\xi}_j) \approx \cos(\boldsymbol{y}_i \angle \boldsymbol{y}_j)$. It is due to this property that our clustering in the reduced space provides a good approximation to the clustering that would be obtained in the unsampled high dimensional space.

This compressive clustering algorithm using complex k-means clustering can be considered a type of sketching algorithm as investigated in [9], where, in our case, we are producing compact vectors that attempt to preserve the original angle between each vector.

4 Performance

We have seen that the iterative process of k-means constantly requires access to the vector data. Therefore, an efficient process would store the vector set in memory to avoid disk accesses. Given this constraint, the size of the data set that we can process depends on the size of the accessible memory.

In this section, we will examine the performance of radial k-means on a small document set from the SMART collection[1]. The document set contains the document set CRAN, CACM, CISI and MED containing 1398, 3204, 1460, and 1033 documents respectively, totalling $7,095$ documents with $14,523$ unique terms.

To reduce the size of the vector space, we ignored all terms that appeared in only one document. By ignoring these terms, we will not affect the results since they do not affect to the similarity between any two documents. This reduced the number of terms from $14,523$ to $7,866$.

4.1 Cluster Accuracy

We computed the accuracy of each cluster using the Jaccard coefficient:

$$J(\mathcal{C}_m, \mathcal{C}_n) = \frac{|\mathcal{C}_m \cap \mathcal{C}_n|}{|\mathcal{C}_m \cup \mathcal{C}_n|}$$

where $\mathcal{C}_m$ and $\mathcal{C}_n$ are the sets of vectors associated to clusters m and n respectively, and $|\cdot|$ is the cardinality operator.

For each experiment performed, we obtained a data set with a recommended clustering. To compute the accuracy, we compared the computed clustering to the recommended clustering and chose the permutation of clusters that optimised the following function:

$$\rho = \underset{p}{\operatorname{argmax}} \left(\sum_{i=1}^{C} J(\mathcal{C}_{p_i}, \mathfrak{C}_i) \right)$$

where $\mathfrak{C}_i$ is the recommended cluster set i, $\mathcal{C}_{p_i}$ is the computed cluster set p_i, p_i is the ith value in permutation p, and ρ is the best matching cluster permutation when compared to the recommended cluster set.

[1] `ftp://ftp.cs.cornell.edu/pub/smart`

4.2 Radial K-means without Sampling

As a baseline measure, we will apply the k-means algorithm to the document vector space without sampling. Doing so produced the results shown in table 1.

Table 1. Cluster accuracy of radial k-means on the whole feature space (no sampling performed). The small size of the document collection allows us to load its contents into memory and perform k-means. We can see that it produces high accuracy, and takes nearly seven minutes to complete.

Accuracy				Time (sec)
CRAN	CACM	CISI	MED	
0.9709	0.9408	0.8798	0.9778	409.82

We can see from the results that radial k-means has successfully partitioned the document collection into the individual document sets, with only the CISI document set accuracy being below 0.9. We can also see that the algorithm took 409.82 seconds to complete (nearly 7 minutes).

4.3 Radial K-means with DFT Sampling

We will now apply the compressive clustering algorithm from section 3.3 to our document collection and compare its accuracy to the clustering found without sampling.

Our first set of results used a sample of the DFT basis as the sampling matrix Φ. We computed results using sample sizes of 16, 32, 64, 128, 256, 512 and 1024. Since the samples are randomly selected we ran 10 trials using each sample size and have reported the mean accuracy in Table 2 and the standard deviation of the accuracy in Table 3.

We can see from the mean accuracy results that there is a clear increase in accuracy as the number of random DFT features used increases, and that the increase saturates at 256 features. We can see that the accuracy for samples of 256 or more are very close to the unsampled clusters obtained in Table 1, being only 0.1 or 0.2 in difference.

Table 2 also contains timing information, showing the mean time taken to run the compressive clustering algorithm. We can see that the times are much shorter than the 409.82 seconds taken without sampling, with 256 DFT features being 14 times faster.

The table of standard deviations (Table 3) provide us with an understanding of the effect of the random sampling. It is interesting to see that the standard deviation is much greater for sampled DFT features of 128 and less when compared to those of 256 and greater. This implies that a low standard deviation in a set of clustering results gives an indication that we have taken enough samples to provide a close approximation to the unsampled clusters. Therefore the cluster standard deviation could easily be used as a measure of the compressive cluster accuracy.

Table 2. Cluster accuracy when using compressive clustering with the Discrete Fourier Transform (DFT). The number of randomly sampled DFT features are provided in the first column, with the accuracy of each cluster (measured using the Jaccard coefficient) in the following four columns, followed by the computation time in seconds.

DFT Feat	Mean Accuracy				Time (sec)
	CRAN	CACM	CICI	MED	
1024	0.9599	0.9196	0.8528	0.9620	74.11
512	0.9258	0.9403	0.8531	0.8944	36.30
256	0.9688	0.9328	0.8607	0.9638	28.49
128	0.8882	0.7117	0.6093	0.5981	24.66
64	0.8219	0.8498	0.7652	0.7184	25.70
32	0.6430	0.7304	0.5976	0.4092	34.93
16	0.7091	0.5815	0.4403	0.3331	24.95

Table 3. Cluster accuracy standard deviation when using compressive clustering with the Discrete Fourier Transform (DFT). The number of randomly sampled DFT features are provided in the first column, with the standard deviation of the accuracy of each cluster (measured using the Jaccard coefficient) in the following four columns.

DFT Feat	Accuracy Standard Deviation			
	CRAN	CACM	CICI	MED
1024	0.0367	0.0670	0.0822	0.0408
512	0.1229	0.0037	0.0660	0.2071
256	0.0028	0.0148	0.0248	0.0132
128	0.1345	0.2112	0.2425	0.3925
64	0.1573	0.1079	0.0912	0.2704
32	0.1844	0.1683	0.1337	0.2357
16	0.1492	0.1412	0.1018	0.1982

4.4 Radial K-means with DCT Sampling

The use of the DFT feature samples requires us to work with complex numbers, meaning that the chosen clustering algorithm also has to cluster complex numbers. Since many clustering clustering algorithms are designed for real numbers only, we will examine a method of sampling real values. Therefore we will examine the effect of DCT feature samples on the accuracy of our compressive clustering algorithm.

Our next set of results used a sample of the DCT basis as the sampling matrix Φ. We computed results using sample sizes of 16, 32, 64, 128, 256, 512 and 1024. Since the samples are randomly selected, we ran 10 trials using each sample size and have reported the mean accuracy in Table 4 and the standard deviation of the accuracy in Table 5.

Table 4. Cluster accuracy when using compressive clustering with the Discrete Cosine Transform (DCT). The number of randomly sampled DCT features are provided in the first column, with the accuracy of each cluster (measured using the Jaccard coefficient) in the following four columns, followed by the computation time in seconds. The superscript dagger (†) denotes a significant difference when compared to the same result using the DFT. The subscript star (∗) denotes a significant difference when compared to the associated result when using the DFT with half the number of features.

DCT Feat	Mean Accuracy				Time (sec)
	CRAN	CACM	CICI	MED	
1024	0.9494	0.8895	0.8022	0.9280	65.10_*
512	$0.9139^\dagger_*$	0.9207	0.8450	0.8977	48.93_*
256	$0.8752^\dagger$	0.8989_*	$0.7804^\dagger$	$0.8006^\dagger$	31.54_*
128	$0.7423^\dagger$	0.7662	0.5652	0.4937	$35.33^\dagger_*$
64	0.7163	0.7442	$0.5867^\dagger$	0.5109	29.78
32	0.5669	$0.5437^\dagger$	$0.4197^\dagger$	0.3207	27.14
16	$0.4694^\dagger$	0.5413	0.4074	0.3019	25.95

From first glance, the accuracy values in Table 4 are lower than those of the DFT in Table 2, but we must remember that shown accuracies are mean values of the ten sample runs for each feature set size. The random sampling of features implies that the results from each experiment will lead to different clustering accuracies. Therefore we employ the use of statistical significance testing to assist us in understanding if there actually is a difference in the clustering, or if the difference is purely by chance.

In each of our experiments, we compute the clustering accuracy obtained using ten different sets of randomly sampled features. Therefore, we have ten different clustering accuracy measurements for each experiment. The statistic that we want to test is if the true mean accuracy of the clustering using the DCT features is different to the true mean accuracy of the clustering using the DFT features.

Since we have only ten samples (ten clustering results) for each method, we cannot have confidence in the t-test, therefore we will employ the use of Wilcoxon rank sum test.

Table 4 contains the results from the significance tests in the form of a superscript dagger (†). A dagger on the accuracy denotes that there is a significant difference at the 0.05 level. We can see that there are four cases that the CRAN cluster is significantly different, three cases where the CISI cluster is significantly different and one case where CACM and MED clusters were significantly different. From examination, we can see that the DCT sampling is worse in all of these cases.

If we consider that the DFT features are complex (containing real and imaginary portions) and that the DCT features are only real, it would be a fairer comparison if we compared the DCT feature results to those of the DFT with half the number of features. For example, if 256 DCT features were selected from a vector, they would occupy the same memory as 128 DFT features, since each DFT feature contains two values (a real and imaginary value).

Table 4 contains the results from the significance tests comparing each DCT feature sample accuracy to each DFT feature sample of half size, in the form of a subscript star ($*$). A star on the accuracy denotes that there is a significant difference at the 0.05 level. We can see that there are now only two cases in total where there is a significant difference, implying that DCT features produce similar results to DFT features when half of the DFT feature samples are used.

Table 4 also contains timing information, showing the mean time taken to run the compressive clustering algorithm. We can see that similar to the DFT sampling, the times are much shorter than the 409.82 seconds taken without sampling, with 512 DCT features being 8 times faster.

Significance tests were also performed on the times to find if the time taken to perform compressive clustering using DCT features was significantly different to the times taken to perform compressive clustering using DFT features. The times with a superscript dagger show a significant difference at the 0.05 level when comparing experiments with the same number of features, while times with subscript star show a significant difference in times at the 0.05 level when comparing experiments using DCT features with those using half the number of DFT features. We can see that there is only one case where a significant difference in time is shown when comparing DCT and DFT with the same number of feature samples. This implies that the use of complex numbers has not increased the time of the clustering. If we examine the subscript stars, we can see that there are four times when there is a significant difference between the time for DCT based compressive clustering and DFT based compressive clustering, where the DCT based method produces longer times.

Table 5. Cluster accuracy standard deviation when using compressive clustering with the Discrete Cosine Transform (DCT). The number of randomly sampled DCT features are provided in the first column, with the standard deviation of the accuracy of each cluster (measured using the Jaccard coefficient) in the following four columns.

DCT Feat	Accuracy Standard Deviation			
	CRAN	CACM	CICI	MED
1024	0.0454	0.1102	0.1640	0.0961
512	0.0699	0.0500	0.0669	0.0972
256	0.1586	0.0952	0.1198	0.2903
128	0.1986	0.1730	0.2540	0.3653
64	0.1975	0.1190	0.2045	0.3012
32	0.1508	0.1626	0.1518	0.2227
16	0.0930	0.1615	0.2737	0.1638

The table of standard deviations (Table 5) provide us with an understanding of the effect of the random sampling. We can see that the table is similar to Table 3 except that the drop in standard deviation does not occur until 512 DCT features are sampled. This re-enforces our belief that the standard deviation can be used to measure the accuracy of our approximation to the unsampled clustering.

5 Clustering Large Scale Document Sets

Our final set of experiments examine the use of compressive clustering on a much larger document set. Just as in the previous section, we have again taken a set of document collections and combined them. The document collections are a set of newspaper articles from the Associated Press (AP), articles from the Financial Review (FR), articles from the Wall Street Journal (WSJ) and articles from Ziff Publishing (ZIFF). These document collections are available on disk two of the TIPSTER collection[2] and were used at TREC-1 to 5. Individually, they contain 79,919, 19,860, 74,520, and 56,920 documents respectively, totalling to 231,219 documents. This document collection contains 208,932 terms. Again, we removed all terms that appeared in only one document, which reduced the term count to 108,734 terms.

Table 6. Cluster accuracy for radial k-means on the whole feature space (no sampling performed) for the large document set (231,219 documents). These results are used as a baseline for the compressive clustering results in Table 7.

Accuracy				Time (sec)
AP	FR	WSJ	ZIFF	
0.6780	0.9962	0.5969	0.8603	181734

We have computed the radial k-means clusters for the large document set (without sampling) as a baseline for the compressive clustering results. The clustering results are shown in Table 6. We can see that the radial k-means process clustered most of the FR and ZIFF documents into correct clusters, but there was confusion amongst the AP and WSJ documents. We can also see that the process took over 50 hours to complete.

To continue the experiment, we ran our compressive clustering algorithm with both the DFT sampling and DCT sampling, and the compressive clustering algorithm. The number of sampled features was chosen as the largest that we could store in memory. For the DFT sampling this was 128 features, while for the DCT features this was 256 features (due to the DFT features being complex and DCT features being real). The recommended clustering for this document

[2] `http://trec.nist.gov/data/docs_eng.html`

Table 7. A comparison of compressive sampling using DFT sampling and DCT sampling. The first two columns provide the sampling transform and the number of features sampled, the following four columns provide the cluster accuracy in terms of the Jaccard coefficient, and the last three columns provide the computation time of the stages involved. Note that the DFT sampling uses half the number of samples since each sample is complex, containing a real and imaginary component.

Transform	Features	Accuracy				Computation Time (sec)			
		AP	FR	WSJ	ZIFF	Pass 1	k-means	Pass 2	Pass 3
DFT	128	0.6704	0.9959	0.5821	0.8617	3626	458	2398	2334
DCT	256	0.6839	0.9964	0.6134	0.8627	3785	1130	2412	2342

collection is each cluster containing only documents from one document set (e.g. cluster 1 contains only documents from the AP document set). The accuracy and timing results are provided in Table 7.

We can see in Table 7 that we obtained high accuracy clusters of the Financial Review and Ziff Publishing articles, implying that the clusters found by k-means was very similar to those in the corresponding collections. The reduction in accuracy for the Associated Press and Wall Street Journal collections shows that there was confusion for a set of the articles as to which cluster they belonged to. The Associated Press and Wall Street Journal collections both contain newspaper articles, therefore there also may be similarity in the content that they each publish. From the results it is obvious that the k-means algorithm has computed similarity between articles from each of the AP and WSJ sources and hence caused a reduction in accuracy.

When comparing the accuracy of DFT and DCT features, Table 7 shows that there is little difference, with the DCT sampled compressive clustering providing slightly greater accuracy, but there is no evidence that one method is better than the other.

The computation time section of Table 7 provides us with a break down of the computation time of each stage in the algorithm. Pass 1 involves scanning through the data set and performing either DFT or DCT feature sampling. The only difference between the two is the time of performing the transform sampling. We can see that the times are similar for both DFT and DCT sampling, implying that the transform sampling times are also the same. The k-means time shows the number of seconds spent computing the radial k-means clusters. It clear that the DFT feature sampled data was much faster to process for k-means than the DCT based data. This is interesting, since as we have said, the DFT features are complex, making the number of integers processed the same as processed in the DCT based features. But the time taken to perform k-means is on the DFT features is less than half of the time to perform k-means on the DCT feature samples.

Pass 2 and 3 involve computing the cluster centres in the unsampled space and recomputing the document assignments to each cluster. We can see that these are both similar and that they are independent of the feature sampling method used.

If we compare the compressive clustering results in Table 7 to the clustering results in Table 6, we find that the accuracy results are very similar. This implies that the compressive clustering algorithm is providing an excellent estimate of the clusters. If we compare the times, we find that the compressive clustering algorithm took approximately 2.5 hours to compute the clusters (for the DFT and DCT methods), while clustering the original data took over 50 hours. This demonstrates the benefit of compressive clustering.

From this we can see that compressive clustering is an exciting new method of computing clusters in high dimensional data. In this article we have explored its application to document clustering, but we can see that with correctly chosen transform and sampling matrices, we can apply compressive clustering to any large scale clustering problem to produce clusters that would have otherwise been not computable.

6 Related Work

The work we have presented is related to Locality-Sensitive Hashing (LSH) [6], and Random projection [2], where in our case the hashing/projection is defined by sampling from the most incoherent linear transformation to the sparse feature space, which in the case of text document vectors, is the Fourier transform or Cosine transform.

Note that when using compressive sampling for dimension reduction and clustering, we are able to reconstruct any vector in the reduced space to its sparse equivalent in the original feature space under certain conditions. This is not possible when performing LSH or Random projection.

7 Conclusion

Clustering large scale data sets requires intense computation and large storage space (such as memory and disk space). The clustering process itself requires continuous access to the data and therefore we benefit from storing the data in memory to avoid lengthy disk accesses.

Compressive sampling is a new concept of Information theory that allows us to sample a small number features from a data set such that we are able to perfectly reconstruct the data, with knowledge of the sampling scheme.

In this article, we presented compressive clustering, an algorithm which utilises the sampling of compressive sampling to obtain a reduced feature space. Using this reduced space, we are able to obtain a close approximation to the clustering that would be obtained on the unsampled data. Using the algorithm presented in this article, compressive clustering can be applied to any domain to obtain approximate clusters in high dimensional data, given appropriate sampling and transform matrices.

We applied compressive clustering using radial k-means to two document collections. We showed that compressive clustering using discrete Fourier and discrete Cosine sampling provided a close approximation to the clusters computed on the unsampled data in 1/14th of the time on the first document set. On the second larger document set containing 231,219 documents, we showed that compressive clustering can provide a very accurate clustering in 1/20th of the time. This shows that compressive clustering is a very useful tool that can be used to quickly compute approximate clusters.

References

1. Baraniuk, R., Davenport, M., DeVore, R., Wakin, M.: A simple proof of the restricted isometry property for random matrices. Constructive Approximation 28, 253–263 (2008), doi:10.1007/s00365-007-9003-x
2. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001, pp. 245–250. ACM, New York (2001)
3. Candes, E.J., Tao, T.: Decoding by linear programming. IEEE Transactions on Information Theory 51(12), 4203–4215 (2005)
4. Candes, E.J., Wakin, M.B.: An introduction to compressive sampling. IEEE Signal Processing Magazine 25(2), 21–30 (2008)
5. Candes, E.J.: Compressive sampling. In: Proceedings of the International Congress of Mathematicians. European Mathematical Society, Madrid (2006)
6. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB 1999, pp. 518–529. Morgan Kaufmann Publishers Inc., San Francisco (1999)
7. Goyal, V.K., Fletcher, A.K., Rangan, S.: Compressive sampling and lossy compression. IEEE Signal Processing Magazine 25(2), 48–56 (2008)
8. Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28(1), 100–108 (1979)
9. Li, P., Church, K.W., Hastie, T.J.: Conditional random sampling: A sketch-based sampling technique for sparse data. In: Advances in Neural Information Processing Systems, vol. 19, p. 873 (2007)