K-means Clustering with Feature Hashing

Hajime Senuma
Department of Computer Science
University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
hajime.senuma@gmail.com

Abstract

One of the major problems of K-means is that one must use dense vectors for its centroids, and therefore it is infeasible to store such huge vectors in memory when the feature space is high-dimensional. We address this issue by using feature hashing (Weinberger et al., 2009), a dimension-reduction technique, which can reduce the size of dense vectors while retaining sparsity of sparse vectors. Our analysis gives theoretical motivation and justification for applying feature hashing to K-means, by showing how much will the objective of K-means be (additively) distorted. Furthermore, to empirically verify our method, we experimented on a document clustering task.

1 Introduction

In natural language processing (NLP) and text mining, clustering methods are crucial for various tasks such as document clustering. Among them, K-means (MacQueen, 1967; Lloyd, 1982) is "the most important flat clustering algorithm" (Manning et al., 2008) both for its simplicity and performance.

One of the major problems of K-means is that it has K centroids which are dense vectors where K is the number of clusters. Thus, it is infeasible to store them in memory and slow to compute if the dimension of inputs is huge, as is often the case with NLP and text mining tasks. A well-known heuristic is truncating after the most significant features (Manning et al., 2008), but it is difficult to analyze its effect and to determine which features are significant.

Recently, Weinberger et al. (2009) introduced feature hashing, a simple yet effective and analyzable dimension-reduction technique for large-scale multitask learning. The idea is to combine features which have the same hash value. For example, given a hash function h and a vector $\boldsymbol{x}$, if $h(1012) = h(41234) = 42$, we make a new vector $\boldsymbol{y}$ by setting $y_{42} = x_{1012} + x_{41234}$ (or equally possibly $x_{1012} - x_{41234}$, $-x_{1012} + x_{41234}$, or $-x_{1012} - x_{41234}$).

This trick greatly reduces the size of dense vectors, since the maximum index value becomes equivalent to the maximum hash value of h. Furthermore, unlike random projection (Achlioptas, 2003; Boutsidis et al., 2010), feature hashing retains sparsity of sparse input vectors. An additional useful trait for NLP tasks is that it can save much memory by eliminating an alphabet storage (see the preliminaries for detail). The authors also justified their method by showing that with feature hashing, dot-product is unbiased, and the length of each vector is well-preserved with high probability under some conditions.

Plausibly this technique is useful also for clustering methods such as K-means. In this paper, to motivate applying feature hashing to K-means, we show the residual sum of squares, the objective of K-means, is well-preserved under feature hashing. We also demonstrate an experiment on document clustering and see the feature size can be shrunk into 3.5% of the original in this case.

Proceedings of the ACL-HLT 2011 Student Session, pages 122–126,
Portland, OR, USA 19-24 June 2011.

2 Preliminaries

2.1 Notation

In this paper, $||\cdot||$ denotes the Euclidean norm, and $\langle\cdot,\cdot\rangle$ does the dot product. $\delta_{i,j}$ is the Kronecker's delta, that is, $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.

2.2 *K*-means

Although we do not describe the famous algorithm of K-means (MacQueen, 1967; Lloyd, 1982) here, we remind the reader of its overall objective for later analysis. If we want to group input vectors into K clusters, K-means can surely output clusters $\omega_1, ...\omega_K$ and their corresponding vectors $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K$ such that they locally minimize the residual sum of squares (RSS) which is defined as

$$\sum_{k=1}^{K} \sum_{\boldsymbol{x}\in\omega_k} ||\boldsymbol{x} - \boldsymbol{\mu}_k||^2.$$

In the algorithm, $\boldsymbol{\mu}_k$ is made into the mean of the vectors in a cluster ω_k. Hence comes the name K-means.

Note that RSS can be regarded as a metric since the sum of each metric (in this case, squared Euclidean distance) becomes also a metric by constructing a 1-norm product metric.

2.3 Additive distortion

Suppose one wants to embed a metric space (X, d) into another one (X', d') by a mapping ϕ. Its additive distortion is the infimum of ϵ which, for any observed $x, y \in X$, satisfies the following condition:

$$d(x, y) - \epsilon \leq d'(\phi(x), \phi(y)) \leq d(x, y) + \epsilon.$$

2.4 Hashing tricks

According to an account by John Langford [1], a co-author of papers on feature hashing (Shi et al., 2009; Weinberger et al., 2009), hashing tricks for dimension-reduction were implemented in various machine learning libraries including Vowpal Wabbit, which he realesed in 2007.

Ganchev and Dredze (2008) named their hashing trick *random feature mixing* and empirically supported it by experimenting on NLP tasks. It is similar to feature hashing except lacking of a binary hash function. The paper also showed that hashing tricks are useful to eliminate alphabet storage.

Shi et al. (2009) suggested *hash kernel*, that is, dot product on a hashed space. They conducted thorough research both theoretically and experimentally, extending this technique to classification of graphs and multi-class classification. Although they tested K-means in an experiment, it was used for classification but not for clustering.

Weinberger et al. (2009) [2] introduced a technique *feature hashing* (a function itself is called the *hashed feature map*), which incorporates a binary hash function into hashing tricks in order to guarantee the hash kernel is unbiased. They also showed applications to various real-world applications such as multitask learning and collaborative filtering. Though their proof for exponential tail bounds in the original paper was refuted later, they reproved it under some extra conditions in the latest version. Below is the definition.

Definition 2.1. Let S be a set of hashable features, h be a hash function $h : S \to \{1, ..., m\}$, and ξ be $\xi : S \to \{\pm 1\}$. The *hashed feature map* $\phi^{(h,\xi)} : \mathbb{R}^{|S|} \to \mathbb{R}^m$ is a function such that the i-th element of $\phi^{(h,\xi)}(\boldsymbol{x})$ is given by

$$\phi_i^{(h,\xi)}(\boldsymbol{x}) = \sum_{j:h(j)=i} \xi(j)x_j.$$

If h and ξ are clear from the context, we simply write $\phi^{(h,\xi)}$ as ϕ.

As well, a kernel function is defined on a hashed feature map.

Definition 2.2. The *hash kernel* $\langle\cdot,\cdot\rangle_\phi$ is defined as

$$\langle\boldsymbol{x}, \boldsymbol{x}'\rangle_\phi = \langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle.$$

They also proved the following theorem, which we use in our analysis.

Theorem 2.3. *The hash kernel is unbiased, that is,*

$$\mathbb{E}_\phi[\langle\boldsymbol{x}, \boldsymbol{x}'\rangle_\phi] = \langle\boldsymbol{x}, \boldsymbol{x}'\rangle.$$

The variance is

$$Var_\phi[\langle\boldsymbol{x}, \boldsymbol{x}'\rangle_\phi] = \frac{1}{m}\left(\sum_{i\neq j} x_i^2 x_j'^2 + x_i x_i' x_j x_j'\right).$$

[1] http://hunch.net/~jl/projects/hash_reps/index.html

[2] The latest version of this paper is at arXiv http://arxiv.org/abs/0902.2206, with correction to Theorem 3 in the original paper included in the Proceeding of ICML '09.

2.4.1 Eliminating alphabet storage

In this kind of hashing tricks, an index of inputs do not have to be an integer but can be any hashable value, including a string. Ganchev and Dredze (2008) argued this property is useful particularly for implementing NLP applications, since we do not anymore need an *alphabet*, a dictionary which maps features to parameters.

Let us explain in detail. In NLP, features can be often expediently expressed with strings. For instance, a feature 'the current word ends with -ing' can be expressed as a string `cur:end:ing` (here we suppose `:` is a control character). Since indices of dense vectors (which may be implemented with arrays) must be integers, traditionally we need a dictionary to map these strings to integers, which may waste much memory. Feature hashing removes this memory waste by converting strings to integers with on-the-fly computation.

3 Method

For dimension-reduction to K-means, we propose a new method *hashed K-means*. Suppose you have N input vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$. Given a hashed feature map ϕ, hashed K-means runs K-means on $\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_N)$ instead of the original ones.

4 Analysis

In this section, we show clusters obtained by the hashed K-means are also good clusters in the original space with high probability. While Weinberger et al. (2009) proved a theorem on (multiplicative) distortion for Euclidean distance under some tight conditions, we illustrate (additive) distortion for RSS. Since K-means is a process which monotonically decreases RSS in each step, if RSS is not distorted so much by feature hashing, we can expect results to be reliable to some extent.

Let us define the difference of the residual sum of squares (DRSS).

Definition 4.1. Let $\omega_1, ...\omega_K$ be clusters, $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K$ be their corresponding centroids in the original space, ϕ be a hashed feature map, and $\boldsymbol{\mu}_1^\phi, ..., \boldsymbol{\mu}_K^\phi$ be their corresponding centroids in the hashed space. Then, DRSS is defined as follows:

$$\begin{aligned} DRSS &= |\sum_{k=1}^{K} \sum_{\boldsymbol{x} \in \omega_k} ||\phi(\boldsymbol{x}) - \boldsymbol{\mu}_k^\phi||^2 \\ &- \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in \omega_k} ||\boldsymbol{x} - \boldsymbol{\mu}_k||^2|. \end{aligned}$$

Before analysis, we define a notation for the (Euclidean) length under a hashed space:

Definition 4.2. The *hash length* $||\cdot||_\phi$ is defined as

$$\begin{aligned} ||\boldsymbol{x}||_\phi &= ||\phi(\boldsymbol{x})|| \\ &= \sqrt{\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}) \rangle} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle_\phi}. \end{aligned}$$

Note that it is clear from Theorem 2.3 that $\mathbb{E}_\phi[||\boldsymbol{x}||_\phi^2] = ||\boldsymbol{x}||^2$, and equivalently $\mathbb{E}_\phi[||\boldsymbol{x}||_\phi^2 - ||\boldsymbol{x}||^2] = 0$.

In order to show distortion, we want to use Chebyshev's inequality. To this end, it is vital to know the expectation and variance of the sum of squared hash lengths. Because the variance of the sum of random variables derives from each covariance between pairs of variables, first we show the covariance between the squared hash length of two vectors.

Lemma 4.3. *The covariance between the squared hash length of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ is*

$$Cov_\phi(||\boldsymbol{x}||_\phi^2, ||\boldsymbol{y}||_\phi^2) = \frac{\psi(\boldsymbol{x}, \boldsymbol{y})}{m},$$

where

$$\psi(\boldsymbol{x}, \boldsymbol{y}) = 2 \sum_{i \neq j} x_i x_j y_i y_j.$$

This lemma can be proven by the same technique described in the Appendix A of Weinberger et al. (2009).

Now we see the following lemma.

Lemma 4.4. *Suppose we have N vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$. Let us define $X = \sum_i ||\boldsymbol{x}_i||_\phi^2 - \sum_i ||\boldsymbol{x}_i||^2 = \sum_i \left(||\boldsymbol{x}_i||_\phi^2 - ||\boldsymbol{x}_i||^2 \right)$. Then, for any $\epsilon > 0$,*

$$P\left(|X| \geq \frac{\epsilon}{\sqrt{m}} \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \psi(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right) \leq \frac{1}{\epsilon^2}.$$

Proof. This is an application of Chebyshev's inequality. Namely, for any $\epsilon > 0$,

$$P\left(|X - \mathbb{E}_\phi[X]| \geq \epsilon\sqrt{Var_\phi[X]}\right) \leq \frac{1}{\epsilon^2}.$$

Since the expectation of a sum is the sum of expectations we readily know the zero expectation: $\mathbb{E}_\phi[X] = 0$.

Since adding constants to the inputs of covariance does not change its result, from Lemma 4.3, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$,

$$Cov_\phi(||\boldsymbol{x}||_\phi^2 - ||\boldsymbol{x}||^2, ||\boldsymbol{y}||_\phi^2 - ||\boldsymbol{y}||^2) = \frac{\psi(\boldsymbol{x}, \boldsymbol{y})}{m}.$$

Because the variance of the sum of random variables is the sum of the covariances between every pair of them,

$$Var_\phi[X] = \frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{N}\psi(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

□

Finally, we see the following theorem for additive distortion.

Theorem 4.5. *Let Ψ be the sum of $\psi(\boldsymbol{x}, \boldsymbol{y})$ for any observed pair of $\boldsymbol{x}, \boldsymbol{y}$, each of which expresses the difference between an example and its corresponding centroid. Then, for any ϵ,*

$$P\left(|DRSS| \geq \epsilon\right) \leq \frac{\Psi}{\epsilon^2 m}.$$

Thus, if $m \geq \gamma^{-1}\Psi\epsilon^{-2}$ where $0 < \gamma <= 1$, with probability at least $1-\gamma$, RSS is additively distorted by ϵ.

Proof. Note that a hashed feature map $\phi^{(h,\xi)}$ is linear, since $\phi(\boldsymbol{x}) = M\boldsymbol{x}$ with a matrix M such that $M_{i,j} = \xi(i)\delta_{h(i),j}$. By this liearlity, $\boldsymbol{\mu}_k^\phi = |\omega_k|^{-1}\sum_{\boldsymbol{x}\in\omega_k}\phi(\boldsymbol{x}) = \phi(|\omega_k|^{-1}\sum_{\boldsymbol{x}\in\omega_k}\boldsymbol{x}) = \phi(\boldsymbol{\mu}_k)$. Reapplying linearlity to this result, we have $||\phi(\boldsymbol{x}) - \boldsymbol{\mu}_k^\phi||^2 = ||\boldsymbol{x} - \boldsymbol{\mu}_k||_\phi^2$. Lemma 4.4 completes the proof. □

The existence of Ψ in the theorem suggests that to use feature hashing, we should remove useless features which have high values from data in advance. For example, if frequencies of words are used as features, function words should be ignored not only because they give no information for clustering but also because their high frequencies magnify distortion.

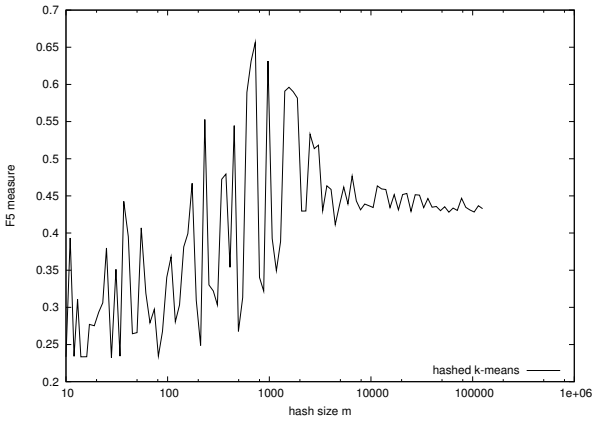

Figure 1: The change of F5-measure along with the hash size

5 Experiments

To empirically verify our method, from 20 Newsgroups, a dataset for document classification or clustering [3], we chose 6 classes and randomly drew 100 documents for each class.

We used unigrams and bigrams as features and ran our method for various hash sizes m (Figure 1). The number of unigrams is 33,017 and bigrams 109,395, so the feature size in the original space is 142,412.

To measure performance, we used the F_5 measure (Manning et al., 2008). The scheme counts correctness pairwisely. For example, if a document pair in an output cluster is actually in the same class, it is counted as true positive. In contrast, if it is actually in the different class, it is counted as false positive. Following this manner, a contingency table can be made as follows:

	Same cluster	Diff. clusters
Same class	TP	FN
Diff. classes	FP	TN

Now, F_β measure can be defined as

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

where the precision $P = TP/(TP+FP)$ and the recall $R = TP/(TP+FN)$.

[3] http://people.csail.mit.edu/jrennie/20Newsgroups/

In short, F_5 measure strongly favors precision to recall. Manning et al. (2008) stated that in some cases separating similar documents is more unfavorable than putting dissimilar documents together, and in such cases the F_β measure (where $\beta > 1$) is a good evaluation criterion.

At the first look, it seems odd that performance can be higher than the original where m is low. A possible hypothesis is that since K-means only **locally** minimizes RSS but in general there are many local minima which are far from the global optimal point, therefore distortion can be sometimes useful to escape from a bad local minimum and reach a better one. As a rule, however, large distortion kills clustering performance as shown in the figure.

Although clustering is heavily case-dependent, in this experiment, the resulting clusters are still reliable where the hash size is 3.5% of the original feature space size (around 5,000).

6 Future Work

Arthur and Vassilvitskii (2007) proposed K-means++, an improved version of K-means which guarantees its RSS is upper-bounded. Combining their method and the feature hashing as shown in our paper will produce a new efficient method (possibly it can be named *hashed K-means++*). We will analyze and experiment with this method in the future.

7 Conclusion

In this paper, we argued that applying feature hashing to K-means is beneficial for memory-efficiency. Our analysis theoretically motivated this combination. We supported our argument and analysis by an experiment on document clustering, showing we could safely shrink memory-usage into 3.5% of the original in our case. In the future, we will analyze the technique on other learning methods such as K-means++ and experiment on various real-data NLP tasks.

Acknowledgements

We are indebted to our supervisors, Jun'ichi Tsujii and Takuya Matsuzaki. We are also grateful to the anonymous reviewers for their helpful and thoughtful comments.

References

Dimitris Achlioptas. 2003. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, June.

David Arthur and Sergei Vassilvitskii. 2007. k-means++ : The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035.

Christos Boutsidis, Anastasios Zouzias, and Petros Drineas. 2010. Random Projections for k-means Clustering. In *Advances in Neural Information Processing Systems 23*, number iii, pages 298–306.

Kuzman Ganchev and Mark Dredze. 2008. Small Statistical Models by Random Feature Mixing. In *Proceedings of the ACL08 HLT Workshop on Mobile Language Processing*, pages 19–20.

Stuart P. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.

J MacQueen. 1967. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and S.V.N. Vishwanathan. 2009. Hash Kernels for Structured Data. *Journal of Machine Learning Research*, 10:2615–2637.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature Hashing for Large Scale Multitask Learning. In *Proceedings of the 26th International Conference on Machine Learning*.