Clustering Large Datasets with Kernel Methods

Stefan Faußer and Friedhelm Schwenker
Institute of Neural Information Processing, University of Ulm, 89069 Ulm, Germany
{stefan.fausser, friedhelm.schwenker}@uni-ulm.de

Abstract
Real-life datasets are becoming larger and less linear separable. Divisive clustering methods with a computation time linear to the number of samples $n$ can handle large data but mostly assume linear boundaries between the cluster in input space. Kernel based clustering methods are able to detect nonlinear boundaries in feature space but have a quadratic computation time $O(n^2)$. In this paper, we propose a meta-algorithm that distributes small-sized subset of the large dataset, parallelized cluster these subset and merges the resulting approximate pseudo-centre repeatedly until the whole dataset has been processed. The meta-algorithm is able to use a wide range of kernel based clustering methods. Here we integrate Kernel Fuzzy C-Means and Relational Neural Gas. We analytically show that the algorithm has a linear computation time $O(n)$. In the experiments we empirically evaluate the performance of the method on two real-life datasets.

1. Introduction
Within the past few years real-life datasets have become significantly larger. Prototype-based clustering algorithms, e.g. k-means, partition the data in input space and have a linear computation time $O(n)$ in terms of the number of samples $n$. However, most of these clustering algorithms assume underlying linear boundaries in the data. Further an euclidean distance between the samples in input space may not be meaningful or even misleading for categorial-type data. Rather a symmetric positive-definite kernel can be selected to match the data features best. A kernel-based clustering method can then partition the data in feature space and is able to detect nonlinear boundaries. Unfortunately these methods have a quadratic computation time and are generally ineligible for large datasets. Further the prototypes solely exist in feature space and are expressed by a linear combination of existing samples.

To cluster large databases, the popular algorithm CLARANS [1] approximates the k-median algorithm by combining sampling techniques with Partitioning Around Medoids (PAM). Later Ester et al. (1996) [2] introduced their density-based clustering algorithm DBSCAN. Compared to CLARANS it does not need the exact number of cluster in advance and detects cluster of arbitrary shapes. Similar to CLARANS two parameters need to be tuned for each dataset. Guha et al. (2003) [3] parallelized the k-median algorithm by dividing the dataset in $l$ data streams and independently clustering them to get $lk$ weighted cluster, represented by samples, and then to recluster the $lk$ samples to get the final $k$ cluster. Guha analytically showed that reclustering of the samples worsens the quantization error and shall therefore be reduced. More recently Fausser and Schwenker (2010) [4] introduced the KPC algorithm that parallelized clusters the data with kernel based methods in small patches and merges $k$ samples for each cluster close to the resulting pseudo-centre. The merged $\mu \cdot K \cdot k$ samples are then weighted by the number of previous hard assigned samples and are reclustered with new samples in the next patch where $\mu$ is the number of parallelizations and $K$ the number of cluster. Very recently Chitta et al. (2011) [5] showed that it is sufficient for Kernel K-Means to define the pseudo-centre as a linear combination of samples from a randomly chosen small subset and to update these pseudo-centre with the rest of the data.

Our contribution in this paper is to generalize the KPC algorithm to calculate and transfer approximate pseudo-centre instead of transferring samples close to the pseudo-centre. We call this algorithm the Parallelized Kernel Patch Clustering with Approximate Pseudo-Centre (KPC-A). The method to select samples to get the approximate pseudo-centre is shown and it is discussed how to set the weights for these approximate solutions. Furthermore, we describe how to parallelized cluster subset of the data using the KPC-A algorithm and how to merge them repeatedly. We show that this algorithm has a linear complexity in terms of
the number of samples. In the experiments we evaluate this meta-algorithm on two real-life datasets namely the NIPS Paper Abstracts and the MiniBooNE dataset. It can be empirically shown that our method KPC-A outperforms the KPC method as well as the basic approach to randomly select samples and to cluster them.

## 2 Approximate Pseudo-Centre

Assume we have randomly sampled $C = \frac{N}{M}$ patches $P_1, \ldots, P_C$ out of $N$ samples, each with a size of $M$. Further suppose we have clustered the first patch $P_1$ in $K$ cluster using a (divisive) kernel based clustering method. The resulting prototypes $c_k$ solely exist in feature space and are defined as a linear combination of assigned samples:

$$
\phi(c_k) = \frac{\sum_{j \in P_1} f_k(x_j) \phi(x_j)}{\sum_{j \in P_1} f_k(j)}
$$

(1)

where $f_k(j)$ is a hard or bounded soft assignment of sample $x_j$ to cluster $c_k$ and $\phi$ is a theoretical mapping function $\phi : X \to \mathbb{F}$, $x_j \in X$, feature space $\mathbb{F}$. Assume we want to recluster the $K$ pseudo-centre with the next patch $P_{t+1}$ to improve the clustering result. Thus we would have to merge all samples from patch $P_1$ and patch $P_{t+1}$ to cluster the previous weighted prototypes $c_k$ formed by a linear combination of the samples from $P_1$ and the new samples in patch $P_{t+1}$. The generalized equation for the prototype $c_k$ with merged previous prototypes is defined as follows:

$$
\phi(c_k') = \frac{\sum_{j \in P_1} f_k(x_j) \phi(x_j)}{\sum_{j \in P_1} f_k(j)} + \frac{\sum_{j \in P_{t+1}} f_k(x_j) \phi(x_j)}{\sum_{j \in P_{t+1}} f_k(j)}
$$

(2)

where $g_k(j)$ are the resulting sample-to-cluster assignments from the last clustering with patch $P_1$ and $w_{i_l}$ are the prototypes weights. If no previous clustering exists, then $g_k(j) = 0, \forall l, \forall j$ and $c_k' = c_k$. Using a symmetric positive-definite kernel $\kappa(x_i, x_j)$ to calculate the similarities between the samples, the distance in feature space between a sample $\phi(x_i)$ and a pseudo-centre $\phi(c_k')$ is as follows:

$$
d(\phi(x_i), \phi(c_k')) = \left\| \phi(x_i) - \phi(c_k') \right\|^2 = \kappa(x_i, x_j) - \frac{\sum_{j \in P_1} f_k(x_j) \phi(x_j)}{\sum_{j \in P_1} f_k(j)} - \frac{\sum_{j \in P_{t+1}} f_k(x_j) \phi(x_j)}{\sum_{j \in P_{t+1}} f_k(j)}
$$

(3)

$$
= \left( \sum_{j \in P_1} f_k(x_j) \phi(x_j) \right)^2 + \left( \sum_{j \in P_{t+1}} f_k(x_j) \phi(x_j) \right)^2 - \left( \sum_{j \in P_1} f_k(x_j) \phi(x_j) \right)^2 - \left( \sum_{j \in P_{t+1}} f_k(x_j) \phi(x_j) \right)^2
$$

$$
= \kappa(x_i, x_j) - \frac{\sum_{j \in P_1} f_k(x_j) \phi(x_j)}{\sum_{j \in P_1} f_k(j)} - \frac{\sum_{j \in P_{t+1}} f_k(x_j) \phi(x_j)}{\sum_{j \in P_{t+1}} f_k(j)}
$$

Analyzing this formula, the kernel matrix increases by each patch, i.e. at step $t$ the matrix has a size of $M \cdot t$ squared. Instead a reduced subset of the samples in $P_t$ may be sufficient to get an approximate pseudo-centre $\hat{c}_k$. The distance between the pseudo-centre $\phi(\hat{c}_k)$ and the approximate pseudo-centre $\phi(c_k')$ is:

$$
d(\phi(\hat{c}_k), \phi(c_k')) = \left\| \phi(\hat{c}_k) - \frac{\sum_{j \in P_t} g_k(i) \phi(x_j)}{\sum_{j \in P_t} g_k(i)} \right\|^2
$$

(4)

where $g_k(i)$ is 1 if sample $x_i$ contributes to the position of the approximate pseudo-centre $\phi(\hat{c}_k)$ or 0 otherwise.

A basic algorithm to select the samples that minimize this distance is shown in 1. It has to be processed for

**Algorithm 1 Fixing Approximate Pseudo-Centre**

**Input**: patch $P_t$, kernel function $\kappa$, cluster $k$, $f_k(i)$ with cluster results of patch $P_t$, prototype weight $w_{ik}$, number of samples per cluster $L$

**Output**: final sample assignments $\tilde{g}_k(i)$

Initialize $\tilde{g}_k(i) = 0, \forall i$

for $l = 1 \to L$
do

Initialize $d_{min} = \infty$

for $i \in P_t$ do

Set $\tilde{g}_k(i) = 1$, calculate $d(\phi(\hat{c}_k), \phi(c_k'))$ with kernel $\kappa$ using equation (4)

if $d(\phi(\hat{c}_k), \phi(c_k')) < d_{min}$ then

$d_{min} = d(\phi(\hat{c}_k), \phi(c_k'))$

$i_{best} = i$

end if

end for

end for

Set $\tilde{g}_k(i) = i_{best}$, remove sample $i_{best}$ from $P_t$

end for
where $\lambda$ is the neighborhood size and 

$$\text{rank}(\phi(e_k), \phi(x_i)) = |\{\phi(e_l) | d(\phi(e_l), \phi(x_i)) < d(\phi(e_k), \phi(x_i)), l = 1, \ldots, K, l \neq k\}| \in \{0, \ldots, K - 1\}. $$

The assignment update step for Kernel Fuzzy C-Means [7] with approximate pseudo-centre is:

$$f_k^\phi(i) = \left(\sum_{l=1}^{K} \left[\frac{d(\phi(e_k), \phi(x_i))}{d(\phi(e_l), \phi(x_i))}\right]^{-\frac{1}{\mu}}\right)^{-1}$$

where $m$ is the fuzzifier. The output of such a kernel based clustering method is then the assignments $f$ that determines the hard (Relational Neural Gas) or soft (Kernel Fuzzy C-Means) assignments of the given samples to $K$ approximate pseudo-centre.

**Algorithm 2** KPC with Approximate Pseudo-Centre

**Input:** samples $x_1, \ldots, x_N$, patch size $M$, kernel function $\kappa$, number of cluster $K$, (divisive) kernel based clustering method, number of samples per cluster $L$

**Output:** prototypes $\phi(e_k)$; final sample-to-cluster assignments $f_k(i)$ and $\tilde{g}_k(i)$, weights $w_k$

Arbitrary distribute $N$ samples in $C = \frac{N}{M}$ patches $P_1, \ldots, P_L$, each with a maximum sample size of $M$.

Initialize $\tilde{g}_k(i) = 0$, $w_k = 1$, $\forall k, \forall i$.

for $t = 1 \rightarrow C$

Calculate kernel matrix using kernel function $\kappa$ with samples of current patch $P_t$ and $L$ K samples $\tilde{g}_k(i) \neq 0, i \in P_{t-1}, \forall k$ from previous patch $P_{t-1}$.

Arbitrary initialize $f \in \mathbb{R}^{K \times (|P_t| + K \cdot L)}$ and apply chosen kernel based clustering method with weights $w$, assignments $\tilde{g}$ and $f$ on patch $P_t$ to get $K$ cluster implicitly defined by sample-to-cluster assignment $f^{new}$.

Determine approximate pseudo-centre for all $K$ cluster, i.e. get $\tilde{g}^{new}$ using weights $w$, $f^{new}$, $\tilde{g}$, and the samples of current patch $P_t$ using algorithm 1.

Update the approximate pseudo-centre weights for all $K$ prototypes using equation (7), set $\tilde{g} = \tilde{g}^{new}$.

end for

3 Parallelized Kernel Patch Clustering with Approximate Pseudo-Centre

With the approximate pseudo-centre introduced in the last section we now propose the complete Kernel Patch Clustering with Approximate Pseudo-Centre algorithm 2. The prototype weights $w_k^l$ are set as follows:

$$w_k^l = \sum_{i \in P_l} f_k^l(i) \cdot (w_k^{l-1} + |e_k^l|)$$

where $f_k^l(i)$ are the resulting sample-to-cluster assignments for patch $P_l$, $w_k^{l-1}$ are the previous prototype weights and $|e_k^l|$ is the number of samples in cluster $k$ without the samples used in the linear combination for the approximate pseudo-centre. Further the prototype weights are normalized by the number of clustered samples, i.e. $w_k^l := \frac{w_k^l \cdot K + \sum_{k=1}^{K} |e_k^l|}{\sum_{k=1}^{K} |e_k^l|}$.

Easily the calculation of the kernel matrix, the clustering process, the fixing of the approximate pseudo-centre and the updating of the prototype weights can be parallelized. For $\mu$ parallelizations, $\mu$ patches are distributed and processed with the main steps in algorithm 2. Then the $\mu \cdot K$ approximate pseudo-centre with their $\mu \cdot K$ prototype weights are redistributed with the next $\mu$ patches until all patches have been processed. The weights are normalized by the number of (last) parallelizations, i.e. $w_k^l := \frac{w_k^l}{\mu}$. The additional approximate pseudo-centre gained by the parallelizations improves the clustering results but also increases the kernel matrix size from $M + K \cdot L$ squared to $M + \mu \cdot K \cdot L$ squared.

The complexity of one clustering operation is $O((M + \mu \cdot K \cdot L)^2)$ where $M$ is the maximum number of samples per patch, $\mu$ the number of parallelizations, $K$ the number of cluster and $L$ the number of samples per cluster. As $M + \mu \cdot K \cdot L$ is bounded and independent on the number of all samples $N$ and we have to perform $\frac{N}{M}$ operations, the summed complexity results in $O((M + \mu \cdot K \cdot L)^2 \cdot \frac{N}{M}) \sim O(M \cdot N) \sim O(N)$ in terms of $N$.

4 Experiments

In the experiments we compared our KPC-A method, the KPC method and the basic approach to cluster a randomly chosen subset of all samples. As for the kernel methods we applied Relational Neural Gas (RNG) and Kernel Fuzzy C-Means (KFCM). In all experiments we set the neighborhood range $\lambda$ for RNG to be exponentially falling from $N/2$ to 0.01. For the internal cluster validation we calculate and compare the quantization error with the resulting cluster to all available samples:

$$E(c_k) = \sum_{k=1}^{K} \sum_{i=1}^{N} f_k(i) \cdot d(\phi(c_k), \phi(x_i))$$

For fair comparisons we measure the distance $d(\phi(c_k), \phi(x_i))$ between $\phi(x_i)$ and the sample nearest to pseudo-centre $\phi(c_k)$ and use hard sample-to-cluster assignments $f$.

The cluster results for the MiniBooNE dataset (see UCI repository [8]) with about 130,000 samples, $K =$
4 cluster, \( L = 25 \) samples per cluster, the RBF-Kernel 
\( \kappa(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2}) \), \( \sigma = 3 \), fuzzifier \( m = 1.25 \) for KFCM, a patch size of \( M = 1,000 \) and 30 test runs can be seen in table 1. For the basic approach (RNG, KFCM) we randomly selected 5,000 samples. The cluster results for the NIPS Paper Abstracts dataset with 1,500 documents, about 12,000 words, \( K = 5 \) cluster, \( L = 15 \) samples per cluster, the Product Probability Kernel with a multinomial distribution (see Jebara et al. (2004) [9]), fuzzifier \( m = 1.05 \) for KFCM, a patch size of \( M = 250 \) and 100 test runs are in the same table 1 (last column). For the basic approach (RNG, KFCM) we randomly selected 500 samples.

<table>
<thead>
<tr>
<th>method</th>
<th>( \mu )</th>
<th>MiniBooNE quant. error</th>
<th>NIPS quant. error</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPC-A (RNG)</td>
<td>1</td>
<td>31952 ± 203</td>
<td>1962 ± 10</td>
</tr>
<tr>
<td>KPC-A (KFCM)</td>
<td>1</td>
<td>32228 ± 446</td>
<td>1960 ± 8</td>
</tr>
<tr>
<td>KPC-A (RNG)</td>
<td>2</td>
<td>31929 ± 811</td>
<td>1960 ± 9</td>
</tr>
<tr>
<td>KPC-A (KFCM)</td>
<td>2</td>
<td>31859 ± 207</td>
<td>1958 ± 8</td>
</tr>
<tr>
<td>KPC (RNG)</td>
<td>1</td>
<td>32046 ± 181</td>
<td>1965 ± 9</td>
</tr>
<tr>
<td>KPC (KFCM)</td>
<td>1</td>
<td>32282 ± 342</td>
<td>1966 ± 12</td>
</tr>
<tr>
<td>KPC (RNG)</td>
<td>2</td>
<td>31952 ± 88</td>
<td>1962 ± 7</td>
</tr>
<tr>
<td>KPC (KFCM)</td>
<td>2</td>
<td>32001 ± 265</td>
<td>1960 ± 8</td>
</tr>
<tr>
<td>RNG</td>
<td>-</td>
<td>32528 ± 246</td>
<td>1969 ± 11</td>
</tr>
<tr>
<td>KFCM</td>
<td>-</td>
<td>32453 ± 225</td>
<td>1964 ± 9</td>
</tr>
</tbody>
</table>

5 Discussing the Results and Conclusion

Analyzing the cluster validation results with the MiniBooNE dataset (table 1), both KPC and KPC-A outperform the basic approach to randomly select 5000 samples and to cluster them although both have only processed \( \leq 1200 \) samples per patch, i.e. \( \leq 200 \) from the last and 1000 samples from the new patch. However our novel KPC-A method further improves the results for one \( (\mu = 1) \) and two \( (\mu = 2) \) parallelizations and for both kernel methods (RNG and KFCM). The results with the NIPS Paper Abstracts dataset confirms this, the KPC-A method combined with KFCM or RNG has a lower quantization error than all other methods have.

We have proposed a novel meta-algorithm KPC-A that performs parallelized clusterings with kernel based methods and calculates, transfers and merges approximate pseudo-centre iteratively until the full (large) dataset is processed. We have empirically shown on two real-life datasets that this method performs better than the KPC method and the basic approach to randomly select a subset of samples and to cluster them. Although we described the integration of two kernel based clustering methods, being the Relational Neural Gas and the Kernel Fuzzy C-Means, further kernel based clustering methods like Self-Organizing Maps can be integrated in the KPC-A algorithm as well.

References