KPCA-based Node Selection for Fast KMSE

Jinghua Wang, Jane You, Qin Li
Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong

ABSTRACT

In this paper we first show that the kernel minimum squared error model is not computationally efficient in feature extraction. To speed up the feature extraction, we linearly express the feature extractor using nodes, i.e. a portion of the training samples in the kernel space. For node selection from the training set, we define two criteria based on Kernel principal component analysis. The nodes are representative and not similar to each other. The experimental results show the feasibility of the proposed method.

Keywords: Minimum Squared Error, Kernel Minimum Squared Error, Feature Extraction, Pattern Recognition

1. INTRODUCTION

Minimum Squared Error (MSE) [1,2] is a popular linear model for feature extraction and classification. The Kernel Minimum Squared Error (KMSE) method [3] is well known for its ability in nonlinear feature extraction. It is proved that KMSE is closely related to Kernel regression, Kernel Fisher discriminant analysis (KFDA) [4], and Support Vector Machine [3]. With the help of Kernel trick, we are able to implicitly map the samples into a high dimension space and transform the nonlinear problem into a linear problem [6,7,8,9]. Specifically, Kernel principal component analysis (KPCA) [15,16] is the nonlinear version of principal component analysis (PCA) [17].

Most kernel methods are proposed based on the Mercer’s theory, i.e. the feature extractor in the kernel space is linearly expressible using the training samples. In the calculation of a single feature, we need to calculate as many kernel functions as the training samples. With this characteristic, the kernel methods are not applicable when the training set is very large. Some methods are proposed to accelerate the feature extraction procedure of the KMSE [5,12,13,14] and KFDA [10,18].

The idea of node is widely used to speed up the nonlinear feature extraction [5,6,10,12,13]. These methods linearly express the feature extractor using a portion of the training samples. The node-based methods work well mainly because some training samples have much larger coefficients than the other samples, i.e. some training samples are more important than the other samples. If we know the important samples, we can linearly express the feature extractor using them.

In this paper, we propose a new method for accelerating the feature extraction procedure of KMSE based on the idea of KPCA. KPCA aims to extract the features by projecting samples on the most representative feature extractors, i.e. the eigenvectors of the covariance matrix. The representative ability of the feature extractors are assessed using the corresponding eigenvalues. In this paper, we select the most representative samples based on the idea of KPCA, and use these representative samples to linearly express the feature extractor. For simplicity, we call these representative samples as nodes. We define two criteria in this paper for node selection: 1) representative ability and 2) similarity. The first criterion assures that all of the nodes are as representative as possible. The second criterion assures that the node set is as small as possible.

Without consideration of the one-time node selection procedure, the proposed method is more computationally efficient than the naïve KMSE: 1) While the naïve KMSE needs to calculate the inverse matrix of size $n \times n$, the proposed method calculate an inverse matrix of size $h \times h$, where $n$ and $h$ are respectively the number of training samples and that of nodes. 2) In order to extract a feature, KMSE and the proposed method respectively need to calculate $n$ and $h$ kernel functions.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the proposed method. Section 4 conducts experiments. Section 5 concludes this paper.

2. RELATED WORK

2.1 Minimum Squared Error (MSE)

For simplicity, in this paper we consider only binary classification problems. The following is the model of MSE

\[
\begin{align*}
X^+ w &= c_1 \\
X^- w &= c_2
\end{align*}
\]

where $X^+ = \begin{bmatrix} x_1^+ & x_2^+ & \cdots & x_m^+ \end{bmatrix}^T$ consists of the positive samples and $X^- = \begin{bmatrix} x_1^- & x_2^- & \cdots & x_m^- \end{bmatrix}^T$ consists of the negative samples. The elements of the vectors $c_1$ and $c_2$ are respectively composed of the class labels of the two classes. Normally, the elements of $c_1$ are all -1 while $c_2$ are all 1.
Denote all the training samples as $X = \begin{bmatrix} X^+ \\ X^- \end{bmatrix}$ and the labels as $C = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$, we can rewrite (1) as

$$Xw = C$$

The equation (1) and (2) can be solved using the least squared-error technique. In a classification problem, we first extract a feature for the sample $x$ using $x^Tw$; then compare $x^Tw$ with $c_1$ and $c_2$. If the distance between $x^Tw$ and $c_1$ is smaller than the distance between $x^Tw$ and $c_2$, we classify the sample $x$ into the positive class; otherwise, we classify it into the negative class.

### 2.2 Kernel Minimum Squared Error (KMSE)

In KMSE, the original sample space is supposed to be mapped into a high dimensional space by a nonlinear function $\phi$. The training samples in the kernel space are $\phi(x_1), \phi(x_2), \cdots, \phi(x_n)$, where $n = n^* + n^-$ is the total number of training samples. Corresponding the nonlinear mapping, there is a kernel function $k(x, y)$ to calculate the inner product of any two mapping samples $\phi(x_i)$ and $\phi(x_j)$ as follows:

$$k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$$

The following is the mathematical model of KMSE [5]:

$$\Phi W = C$$

where

$$W = \begin{bmatrix} w_0 \\ w \end{bmatrix}, C = \begin{bmatrix} 1 \\ \vdots \\ -1 \end{bmatrix}, \Phi = \begin{bmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{bmatrix}$$

The feature extractor $W$ consists of a threshold $w_0$ and the vector $w$ in the kernel space. Note, the dimensionality of $w$ equals that of the training samples $\phi(x_1), \phi(x_2), \cdots, \phi(x_n)$. The element $C_i(1 \leq i \leq n)$ is the class label of the $i$th sample $\phi(x_i)(1 \leq i \leq n)$. KMSE can be considered as a machine, where $\phi(x_i)$ is the input and its label $C_i$ is the output. Based on the Mercer’s theory, we can express the feature extractor as follows [3]:

$$W = \begin{bmatrix} w_0 \\ \sum_{i=1}^n \alpha_i \phi(x_i) \end{bmatrix}$$

Substituting equation (6) into equation (4), we obtain:

$$KA = C$$

where

$$A = \begin{bmatrix} w_0 \\ \alpha_i \end{bmatrix}, K = \begin{bmatrix} 1 & k(x_i, x_j) & \cdots & k(x_i, x_n) \\ k(x_2, x_j) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_j) & \cdots & k(x_n, x_n) \end{bmatrix}$$

Equation (9) is the normal equation of (7) and they have the same solution:

$$\begin{bmatrix} K^\top K \end{bmatrix}A = K^\top C$$

If the matrix $K$ is a full column rank matrix, the solution is:

$$A = (K^\top K)^{-1} K^\top C$$

If the matrix $K^\top K$ is singular, equation (11) calculates a numerical stable solution of equation (7):

$$A = (K^\top K + \mu I)^{-1} K^\top C$$

where $\mu$ is a positive constant and $l$ is the identity matrix [16].

For a test sample $x$, KMSE extracts a feature by projecting its mapping sample $\phi(x)$ onto the feature extractor $W$, as follows

$$l_\phi(x) = w_0 + \sum_{i=1}^n \alpha_i k(x, x_i)$$

If $l_\phi(x) > 0$, we label $x$ with 1; or else, we label it with $-1$.

### 2.3 Kernel Principal Component Analysis (KPCA)

Here, we assume the mapping samples are centered. Then, we can calculate the covariance matrix as follows

$$S_\phi^\nu = \sum_{i=1}^n \phi(x_i)\phi(x_i)^\top = XX^\top$$

where $X = \begin{bmatrix} \phi(x_1) & \phi(x_2) & \cdots & \phi(x_n) \end{bmatrix}$ is a matrix consists of all the mapping samples. KPCA aims to extract the most representative features and the feature extractors are the eigenvectors of the following eigenequation

$$S_\phi^\nu \nu = \lambda \nu$$

The importance of a feature extractor is measured by the corresponding eigenvalue. To achieve the goal of dimension reduction, we usually only keep the most important feature extractors. These feature extractors are linearly expressed by all of the training samples [11], i.e.

$$\nu = \sum_{i=1}^n \alpha_i \phi(x_i) = X \alpha$$

where $\alpha = [\alpha_1 \alpha_2 \cdots \alpha_n]^\top$ is a vector in $n$ dimensional space. With equation (14) and (15), we obtain the flowing matrix

$$PP\alpha = \lambda P\alpha \Rightarrow P\alpha = \lambda \alpha$$

where the matrix $P = XX^\top$ is defined to be

$$P_{ij} = k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)(1 \leq i, j \leq n).$$
3. PROPOSED METHOD

As can be seen from equation (12), we needs to calculate the kernel function between $x$ and every training sample. Thus, the feature extraction efficiency of KMSE is inversely proportional to the size of the training set. This means KMSE is not applicable in large scale problems. It is necessary to improve KMSE for fast feature extraction.

In the proposed method, we assume the feature extractor of KMSE can be linearly approximated by nodes, i.e. a portion of important training samples. We have such an assumption based on the observation that some training samples have much larger coefficient than the rest. To approximate the feature extractor as well as possible, the nodes should be as representative as possible. If we can use these nodes to linearly approximate the training samples which are not nodes, we can use the node set to replace the training set in the linear expression of the feature extractor. Before presenting the criteria for node section 3.2, we first show the formulation of the proposed method in section 3.1.

3.1 The proposed method

Suppose the feature extractor can be approximated as follows

$$W \approx W' = \begin{bmatrix} w_0 \\ \sum_{i=1}^h \beta_i \phi(x'_i) \end{bmatrix}$$  \hspace{1cm} (17)

where $x'_1, x'_2, \ldots, x'_h$ are the nodes and $\beta = [\beta_1, \beta_2, \ldots, \beta_h]^T$ is a $h$ by 1 vector. In the following, we replace the sign “≈” in (17) with “=”. By substituting (17) into (4), we obtain

$$K' A' = C$$  \hspace{1cm} (18)

where

$$A' = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_h \end{bmatrix}, \quad K' = \begin{bmatrix} k(x_1, x'_1) & \cdots & k(x_1, x'_h) \\ k(x_2, x'_1) & \cdots & k(x_2, x'_h) \\ \vdots & \vdots & \vdots \\ k(x_n, x'_1) & \cdots & k(x_n, x'_h) \end{bmatrix}$$  \hspace{1cm} (19)

A similar equation to (11) is used to calculate the coefficient vector $A'$, as follows

$$A = \left( K'^T K' + \mu I \right)^{-1} K'^T C$$  \hspace{1cm} (20)

For a test sample $x$, the proposed method extracts a feature by projecting its mapping sample $\phi(x)$ onto the feature extractor $W'$, as follows

$$l'_x(x) = w_0 + \sum_{i=1}^h \beta_i k(x, x'_i)$$  \hspace{1cm} (21)

If $l'_x(x) > 0$, we label $x$ with $1$; or else, we label it with $-1$.

In (21), we only needs to calculate a kernel function for each of nodes. In our experiments, the nodes are only a small portion of the training samples. Thus, the feature extraction procedure in (21) in much more computationally efficient than that in (12). In addition, the equation (20) calculate an inverse matrix of a matrix of size $h$ by $h$. Differently, equation (11) has to calculate an inverse matrix of size $n$ by $n$. So, the calculation of the coefficient vector in the proposed method is more computationally efficient.

3.2 Node selection

The performance of the proposed method depends on the node set. Generally speaking, the more representative the node set is, the better we can use the nodes to approximate the feature extractor. Here, we employ the idea of KPCA. In KPCA, the importance of an eigenvector is assessed using the corresponding the eigenvalue. Similarly, we define a criterion (pseudo-eigenvalue) $l$ to assess the training sample $\phi(x)$ as follows

$$l = \frac{\phi^T(x) S_n^p \phi(x)}{\phi^T(x) \phi(x)}$$  \hspace{1cm} (22)

The larger the value $l$, the more representative the training sample is. While a matrix $S_n^p$ only have $n$ eigenvalues, it has a pseudo-eigenvalue for each of the training sample.

In order to reduce the size of the node set, we bound the similarity between the select nodes using the following criteria

$$\cos(x_i, x_j) = \frac{\phi^T(x_i) \phi(x_j)}{\sqrt{\phi^T(x_i) \phi(x_i)} \sqrt{\phi^T(x_j) \phi(x_j)}}$$  \hspace{1cm} (23)

If a sample is selected as a node, all the other samples which are similar to it is deleted from the candidate set.

The following is the node selection algorithm

- Step 1 initialize the node set to be null and the candidate set to be the training set;
- Step 2 put the most representative sample into
the node set;
- Step 3 delete each sample from the candidate set if its similarity with the node is higher than a threshold;
- Step 4 if the candidate set is not null, go to step 2.

With the selected node, we can formulate the proposed method according to (18) and extract features using (21).

4. EXPERIMENTS

Our experiments are conducted using several benchmark datasets downloaded from http://archive.ics.uci.edu/ml. For each dataset, we randomly separate it into training set and testing set and repeat ten times. We use the Gaussian kernel
\[ k(x, y) = \exp\left(-\|x - y\|^2 / 2\sigma^2\right). \]
The parameter \( \sigma^2 \) is automatically set to be the sum of the variances of all the data components in the training sample subset. The parameter \( \mu \) equals to 0.1 for all the datasets. We conduct experiments using the ten divisions of the sample set. The average of ten classification accuracy is regarded as final accuracy in our experiments.

We compare the proposed method with the standard KMSE and its three improvements, RKMSE[12], DKMSE[13], MPLOC-KMSE[14], and EKMSE[5]. All of the four methods speed up the feature extraction procedure by the idea of node, i.e. expressing the feature extractor using a portion of the training samples.

<table>
<thead>
<tr>
<th>dataset</th>
<th>Titanic</th>
<th>flare</th>
<th>Heart</th>
<th>German</th>
</tr>
</thead>
<tbody>
<tr>
<td>(number of samples)</td>
<td>150</td>
<td>666</td>
<td>170</td>
<td>700</td>
</tr>
<tr>
<td>RKMSE[12]</td>
<td>26</td>
<td>49</td>
<td>64</td>
<td>238</td>
</tr>
<tr>
<td>DKMSE[13]</td>
<td>17</td>
<td>51</td>
<td>70</td>
<td>180</td>
</tr>
<tr>
<td>MPLOC-KMSE[14]</td>
<td>7</td>
<td>14</td>
<td>90</td>
<td>206</td>
</tr>
<tr>
<td>EKMSE[5]</td>
<td>8</td>
<td>19</td>
<td>47</td>
<td>186</td>
</tr>
<tr>
<td>Proposed method</td>
<td>5</td>
<td>9</td>
<td>39</td>
<td>169</td>
</tr>
</tbody>
</table>

5. CONCLUSION

MSE and KMSE are extensively used in application. However, the feature extraction procedure of the KMSE is inversely proportional to the size of the training set. In this paper, we reformulate the KMSE by linearly express the feature extractor as a linear combination of a portion of the training set. This speeds up the calculation of the coefficient vector as well as the feature extraction procedure. The experimental results show the feasibility of the proposed method.

6. REFERENCES