# Inverse Matrix Update for Kernel Matrix

Shuo Jiang

In this document, we summarized 3 different inverse matrix update methods for kernel matrixes. The methods avoid direct inversion when some new datasets are added. The three methods can be used in different situations.

## Symbols

$a$ : a scalar

$a_i$ :a scalar at index $i$

$x_{i:j}$ :a subset of dataset between index $i$ and index $j$, including $x_i$ and $x_j$

$\vec{a}$ : a column vector

$\vec{a}^T$ : a row vector

$\mathbf{A}$ : a matrix

$\mathbf{A}(:, j)$ : the $j^{th}$ column of matrix $\mathbf{A}$

$\mathbf{A}(i,:)$ : the $i^{th}$ row of matrix $\mathbf{A}$

## 1. Kernel Matrix

Kernel methods are widely used in machine learning field (Support Vector Machine, Gaussian Process).

The kernel matrix $\mathbf{K}$ is generally formed by a dataset $X = \left\{\vec{x_1}, \vec{x_2}, \cdots, \vec{x_n}\right\}$.

The entries of $\mathbf{K}$ is determined by kernel function

$$k\left(\vec{x_i}, \vec{x_j}\right). \tag{1}$$

Typical kernel functions are, for example,
Squared Exponential kernel function:

$$k\left(\vec{x_i}, \vec{x_j}\right) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\left(\vec{x_i} - \vec{x_j}\right)^2\right) + \sigma_n^2 \delta_{xixj}$$

The Matern class of kernel function:

$$k(r) = \frac{2^{1-\upsilon}}{\Gamma(\upsilon)}\left(\frac{\sqrt{2\upsilon}r}{l}\right)^{\upsilon} K_{\upsilon}\left(\frac{\sqrt{2\upsilon}r}{l}\right)$$

with $r = \left| \vec{x_i} - \vec{x_j} \right|$ and $\Gamma\left(\vec{x}\right)$ is gamma function, where $K_\upsilon\left(x\right)$ is modified Bessel function.

Then form of kernel matrix is like:

$$\mathbf{K} = \begin{bmatrix} k\left(\vec{x_1}, \vec{x_1}\right) & k\left(\vec{x_1}, \vec{x_2}\right) & \cdots & k\left(\vec{x_1}, \vec{x_n}\right) \\ k\left(\vec{x_2}, \vec{x_1}\right) & k\left(\vec{x_2}, \vec{x_2}\right) & \cdots & k\left(\vec{x_2}, \vec{x_n}\right) \\ \vdots & \vdots & \ddots & \vdots \\ k\left(\vec{x_n}, \vec{x_1}\right) & k\left(\vec{x_n}, \vec{x_2}\right) & \cdots & k\left(\vec{x_n}, \vec{x_n}\right) \end{bmatrix} \tag{2}$$

Usually, by properly choosing kernel function, the kernel matrix is positive definite and symmetrical. Means:

$$\mathbf{K^T} = \mathbf{K} \tag{3}$$

$$\vec{x}^T \cdot \mathbf{K} \cdot \vec{x} > 0 \quad \text{for all} \quad \vec{x}$$

## 2. Matrix Inversion

In some cases, the inversion of kernel matrix is needed. The most common method is using the Cholesky decomposition [1] to do the inversion.

Every symmetric, positive definite matrix $\mathbf{A}$ can be decomposed into a product of a unique lower triangular matrix $\mathbf{L}$ and its transpose as $\mathbf{A} = \mathbf{L}\mathbf{L}^\mathbf{T}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \mathbf{L}\mathbf{L}^T = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ 0 & l_{22} & \cdots & l_{n2} \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & l_{nn} \end{bmatrix} \tag{4}$$

the entries of $\mathbf{L}$ can be determined iteratively from $l_{11}$ as

$$l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$$

$$\text{and} \quad l_{ik} = \frac{1}{l_{kk}} \left( a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj} \right) \tag{5}$$

The inversion of matrix $\mathbf{A}$ can be calculated as [2]:

$$\mathbf{A}^{-1} = \left(\mathbf{L}^{-1}\right)^T \left(\mathbf{L}^{-1}\right) \tag{6}$$

if we denote $\mathbf{L}^{-1}=\mathbf{H}$, so $\mathbf{LH}=\mathbf{I}$, $\mathbf{I}$ is unit matrix.

$$\mathbf{LH}=\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \vdots \\ \vdots & & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n1} & h_{n2} & \cdots & h_{nn} \end{bmatrix}=\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

so by expanding the equation at left for each row of $\mathbf{L}$, we get expressions like:

$$\begin{cases} l_{11}h_{11}=1 \\ l_{11}h_{12}=0 \\ l_{11}h_{13}=0 \\ l_{11}h_{14}=0 \\ \vdots \end{cases} \qquad \begin{cases} l_{21}h_{11}+l_{22}h_{21}=0 \\ l_{21}h_{12}+l_{22}h_{22}=1 \\ l_{21}h_{13}+l_{22}h_{23}=0 \\ l_{21}h_{14}+l_{22}h_{24}=0 \\ \vdots \end{cases} \qquad \begin{cases} l_{31}h_{11}+l_{32}h_{21}+l_{33}h_{31}=0 \\ l_{31}h_{12}+l_{32}h_{22}+l_{33}h_{32}=0 \\ l_{31}h_{13}+l_{32}h_{23}+l_{33}h_{33}=1 \\ l_{31}h_{14}+l_{32}h_{24}+l_{33}h_{34}=0 \\ \vdots \end{cases} \quad \cdots\cdots$$

(a) (b) (c)

As $l_{ij}$ are known, in set (a), we can calculate the value o f the first row $h_{1j}$ of $\mathbf{H}$, and this result can be used to calculate the second set (b) as the second row $h_{2j}$. Then we can calculate the value of $\mathbf{H}$ iteratively by this method. Finally, as defined $\mathbf{L}^{-1}=\mathbf{H}$, we can use equation (6) to calculate $\mathbf{A}^{-1}$.

## 3. Augmented Inverse Update

When some samples have to be added to dataset $X$ or deleted from $X$, the corresponding kernel matrix and its inverse should be recalculated. For kernel matrix, we only have to change the value of one row and one column, so this update is efficient. However when an element of $X$ is changed , the whole inverse matrix $\mathbf{K}^{-1}$ is completely changed. It can be proven that the direct inversion in last chapter requires $O\left(n^3\right)$ complexity, which can be unacceptable in many practical cases. So, alternative calculation without direct inversion should be invented with less complexity. In some special cases, the update with less complexity of inverse kernel matrix exists.

The First Method is called Augmented Update [3].
It corresponds to that we already have known a dataset $X$ and the kernel matrix $\mathbf{K}$ it forms by (1) as:

$$X=\begin{bmatrix}\overrightarrow{x_1}\\ \overrightarrow{x_2}\\ \vdots\\ \overrightarrow{x_n}\end{bmatrix} \quad \Longrightarrow \quad \mathbf{K}=\begin{bmatrix} k\left(\overrightarrow{x_1},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_1},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_1},\overrightarrow{x_n}\right)\\ k\left(\overrightarrow{x_2},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_2},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_2},\overrightarrow{x_n}\right)\\ \vdots & \vdots & \ddots & \vdots\\ k\left(\overrightarrow{x_n},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_n},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_n},\overrightarrow{x_n}\right)\end{bmatrix}$$

Also, the $\mathbf{K}^{-1}$ has been calculated by the matrix inversion method in last chapter.
When a new sample is added to the end of the dataset $X$ as

$$X=\begin{bmatrix}\overrightarrow{x_1}\\ \overrightarrow{x_2}\\ \vdots\\ \overrightarrow{x_n}\\ \overrightarrow{x_{n+1}}\end{bmatrix}$$ and we want to calculate the new kernel matrix.

In order to distinguish these two situations, we denote the matrix formed by $X$ of $n$ samples as $\mathbf{K}_n$, and its inverse $\mathbf{K}_n^{-1}$. The kernel matrix formed by $X$ of $n+1$ samples as $\mathbf{K}_{n+1}$ and $\mathbf{K}_{n+1}^{-1}$. As we have mentioned, $\mathbf{K}_n$ and $\mathbf{K}_n^{-1}$ are known. $\mathbf{K}_{n+1}$ can be calculated with adding a new row and column at the end of $\mathbf{K}_n$ as

$$\mathbf{K}_{n+1}=\left[\begin{array}{cccc|c} k\left(\overrightarrow{x_1},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_1},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_1},\overrightarrow{x_n}\right) & k\left(\overrightarrow{x_1},\overrightarrow{x_{n+1}}\right)\\ k\left(\overrightarrow{x_2},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_2},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_2},\overrightarrow{x_n}\right) & k\left(\overrightarrow{x_2},\overrightarrow{x_{n+1}}\right)\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ k\left(\overrightarrow{x_n},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_n},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_n},\overrightarrow{x_n}\right) & k\left(\overrightarrow{x_n},\overrightarrow{x_{n+1}}\right)\\ \hline k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_n}\right) & k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{n+1}}\right)\end{array}\right]=\begin{bmatrix}\mathbf{K}_n & \overrightarrow{k\left(\overrightarrow{x_i},\overrightarrow{x_{n+1}}\right)}\\ \overrightarrow{k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_j}\right)}^T & k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{n+1}}\right)\end{bmatrix} \tag{7}$$

This operation is very efficient, however if we want to calculate $\mathbf{K}_{n+1}^{-1}$ using the inversion method mentioned in last chapter, it can be inefficient as $n$ is large.

We can express the relation in a compact way

$$\mathbf{K}_{n+1}=\begin{bmatrix}\mathbf{K}_n & \vec{b}\\ \vec{b}^T & d\end{bmatrix} \tag{8}$$

as the same form as in (7). $\vec{b}$ is sub-matrix of size $n\times 1$ calculated as the kernel function values between each $x_i$ in original dataset and the new sample $x_{n+1}$. $d$ is an element of kernel function value of $x_{n+1}$ and itself.

The solution of updated $\mathbf{K}_{n+1}^{-1}$ is

$$\mathbf{K}_{n+1}^{-1}=\begin{bmatrix}\mathbf{K}_n^{-1}+g\vec{e}\cdot\vec{e}^{T} & -g\vec{e}\\ -g\vec{e}^{T} & g\end{bmatrix} \tag{9}$$

with $\vec{e}=K_n^{-1}\cdot\vec{b}$ and $g=\left(d-\vec{b}^{T}\cdot\vec{e}\right)^{-1}$.

A demonstration of this is

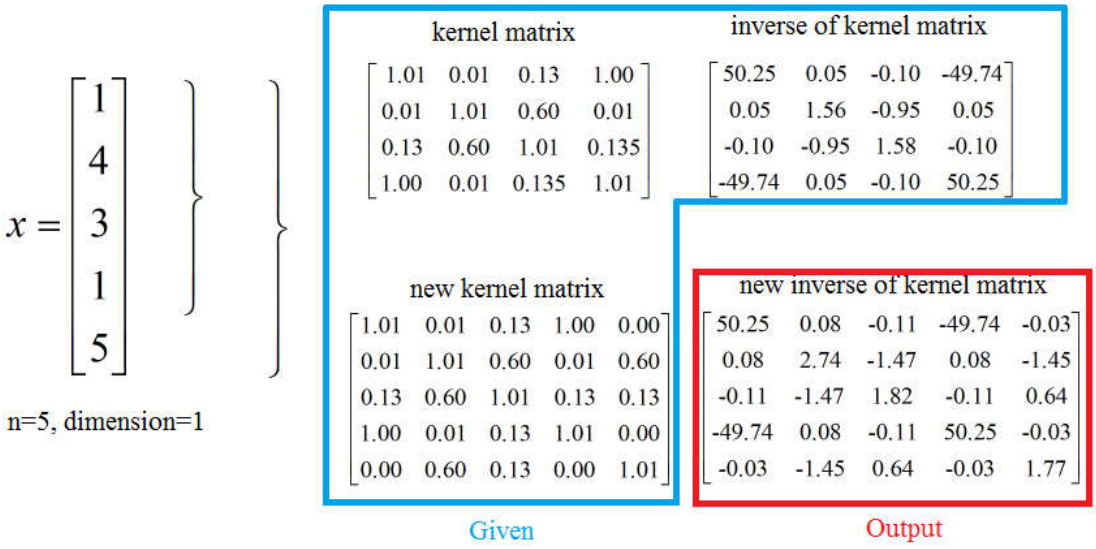


Figure.1 Demonstration of Augmented Update

The kernel matrix and inverse of it in the upper part of figure is calculated from the "old" kernel as the first 4 elements of $X$. As the new sample "5" is added in the dataset, the kernel matrix is augmented for an additional dimension. The matrix enclosed by blue is matrixes known or easy to be calculated, and the matrix enclosed by red is the result of (9) as the new kernel matrix $\mathbf{K}_{n+1}^{-1}$.

The kernel value is calculated using exponential square kernel with $\sigma_f=1$, $l=1$ and $\sigma_n=0.1$.

It can be proven that this update only takes $O\left(n^2\right)$ complexity. This method will not loss any accuracy while being efficient, because the direct inversion causes a lot redundant calculation.

## 4. Sliding Window Update

The next situation [4] is when a new sample is added at the end of $X$, however we want to keep the size of the dataset unchanged, so we eliminate the first element in this dataset. This operation is like a FIFO system.

$$X=\begin{bmatrix}\overrightarrow{x_1}\\ \overrightarrow{x_2}\\ \vdots\\ \overrightarrow{x_n}\end{bmatrix} \quad \Longrightarrow \quad X=\begin{bmatrix}\overrightarrow{x_2}\\ \vdots\\ \overrightarrow{x_n}\\ \overrightarrow{x_{n+1}}\end{bmatrix}$$

As the kernel matrixes of these two datasets are both of size $n\times n$, in this case, we use $\mathbf{K}_{old}$, $\mathbf{K}_{old}^{-1}$ to denote the kernel matrix and its inverse before update, and $\mathbf{K}_{new}, \mathbf{K}_{new}^{-1}$ for the updated dataset.

$$\mathbf{K}_{old}=\left[\begin{array}{c|ccc} k\left(\overrightarrow{x_1},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_1},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_1},\overrightarrow{x_n}\right)\\ \hline k\left(\overrightarrow{x_2},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_2},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_2},\overrightarrow{x_n}\right)\\ \vdots & \vdots & \ddots & \vdots\\ k\left(\overrightarrow{x_n},\overrightarrow{x_1}\right) & k\left(\overrightarrow{x_n},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_n},\overrightarrow{x_n}\right)\end{array}\right]=\begin{bmatrix}a & \vec{b}^T\\ \vec{b} & \mathbf{D}\end{bmatrix} \tag{10}$$

$a$ is the first element of $\mathbf{K}_{old}$, $\vec{b}$ is the sub-matrix at the first column of size $(\mathrm{n}-1)\times 1$. $\mathbf{D}$ is the sub-matrix at the right bottom corner of size $(\mathrm{n}-1)\times(\mathrm{n}-1)$.

Also we have known the inverse of "old" kernel matrix

$$\mathbf{K}_{old}^{-1}=\begin{bmatrix}e & \vec{f}^T\\ \vec{f} & \mathbf{H}\end{bmatrix} \tag{11}$$

The segmentation is similar to (10).

The kernel matrix of the updated dataset is

$$\mathbf{K}_{new}=\left[\begin{array}{ccc|c} k\left(\overrightarrow{x_2},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_2},\overrightarrow{x_n}\right) & k\left(\overrightarrow{x_2},\overrightarrow{x_{n+1}}\right)\\ \vdots & \ddots & \vdots & \vdots\\ k\left(\overrightarrow{x_n},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_n},\overrightarrow{x_n}\right) & k\left(\overrightarrow{x_n},\overrightarrow{x_{n+1}}\right)\\ \hline k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_2}\right) & \cdots & k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_n}\right) & k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{n+1}}\right)\end{array}\right]=\begin{bmatrix}\mathbf{D} & \vec{h}\\ \vec{h}^T & g\end{bmatrix} \tag{12}$$

$\mathbf{D}$ is the same as in (10), $g$ is the last element of $\mathbf{K}_{new}$, $\vec{h}$ is the sub-matrix at the first column of size $(\mathrm{n}-1)\times 1$.

Then, with the help of $\mathbf{K}_{old}, \mathbf{K}_{old}^{-1}$ and $\mathbf{K}_{new}$, we can calculate $\mathbf{K}_{new}^{-1}$ as

$$\mathbf{K}_{new}^{-1}=\begin{bmatrix}\mathbf{B}+\left(\mathbf{B}\cdot\vec{h}\right)\left(\mathbf{B}\cdot\vec{h}\right)^T s & -\left(\mathbf{B}\cdot\vec{h}\right)s\\ -\left(\mathbf{B}\cdot\vec{h}\right)^T s & s\end{bmatrix} \tag{13}$$

with $\mathbf{B} = \mathbf{H} - \frac{\vec{f} \cdot \vec{f}^T}{e}$ , $s = \frac{1}{g - \vec{h}^T \cdot \mathbf{B} \cdot \vec{h}}$.

A demonstration is

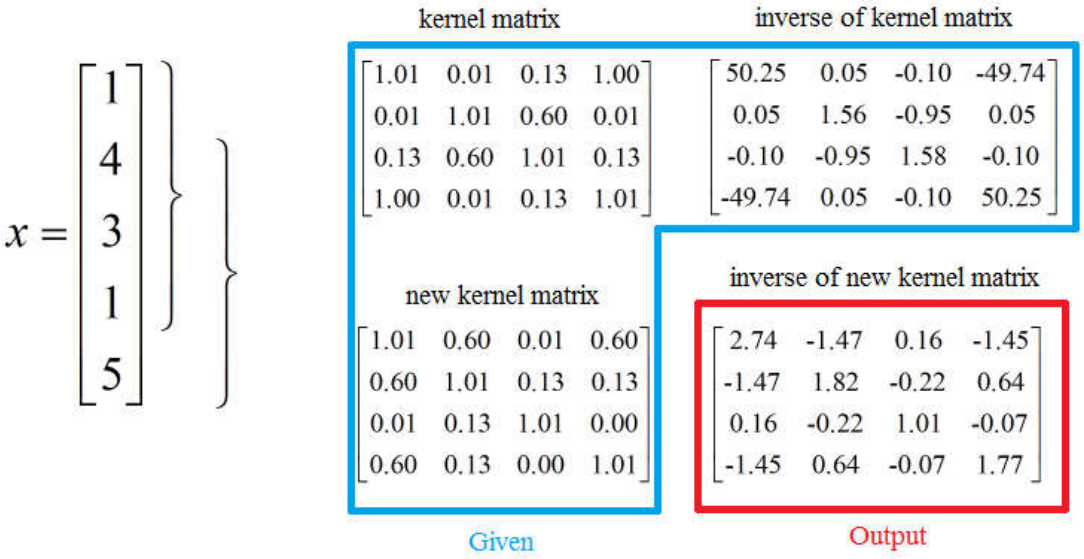


Figure.2 Demonstration of Sliding Window Update

The kernel matrix and inverse of it in the upper part of figure is calculated from the "old" kernel as the first 4 elements of $X$. As the new sample "5" is added in the dataset, the first element "1" is eliminated to keep the size of kernel matrix size constant. The matrix enclosed by blue is matrixes known or easy to be calculated, and the matrix enclosed by red is the result of (13) as the new kernel matrix $\mathbf{K}_{new}^{-1}$.

It can be proven that this inverse update only takes $O(n^2)$ complexity.

## 5. Sherman-Morrison Update

The next situation is more general, where we have the original dataset $X$, one of its element is replaced by a new sample [3, 5].

$$X = \begin{bmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_j \\ \vdots \\ \vec{x}_n \end{bmatrix} \Longrightarrow X = \begin{bmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_{n+1} \\ \vdots \\ \vec{x}_n \end{bmatrix}$$

(a) (b)

The old kernel matrix formed by (a) is like (2), when the element at index $j$ is replaced, only the

$j^{th}$ row and $j^{th}$ column are changed.

The updated kernel matrix can be expressed in such form

$$\mathbf{K}_{new}=\begin{bmatrix} \mathbf{K}_{old}(a) & \vec{k}\left(\overrightarrow{x_{1:j-1}},\overrightarrow{x_{n+1}}\right) & \mathbf{K}_{old}(b) \\ \vec{k}^{T}\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{1:j-1}}\right) & k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{n+1}}\right) & \vec{k}^{T}\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{j+1:n}}\right) \\ \mathbf{K}_{old}(c) & \vec{k}\left(\overrightarrow{x_{j+1:n}},\overrightarrow{x_{n+1}}\right) & \mathbf{K}_{old}(d) \end{bmatrix} \tag{14}$$

where $\mathbf{K}_{old}(a,b,c,d)$ are sub-matrix of $\mathbf{K}_{old}$, $\mathbf{K}_{old}(a)$ is the sub-matrix whose rows are $1$ to $j-1$ and columns are $1$ to $j-1$ of $\mathbf{K}_{old}$. $\mathbf{K}_{old}(b)$ is the sub-matrix whose rows are $1$ to $j-1$ and columns are $j+1$ to $n$ of $\mathbf{K}_{old}$. $\mathbf{K}_{old}(c)$ is the sub-matrix whose rows are $j+1$ to $n$ and columns are $1$ to $j-1$ of $\mathbf{K}_{old}$. $\mathbf{K}_{old}(d)$ is the sub-matrix whose rows are $j+1$ to $n$ and columns are $j+1$ to $n$ of $\mathbf{K}_{old}$. $\vec{k}\left(\overrightarrow{x_{1:j-1}},\overrightarrow{x_{n+1}}\right)$ is a vector of size $(n-1)\times 1$, whose elements are the kernel function value between $\overrightarrow{x_i}$ in range $1$ to $j$ and new sample $x_{n+1}$. Similar explanations with $\vec{k}\left(\overrightarrow{x_{j+1:n}},\overrightarrow{x_{n+1}}\right)$, $\vec{k}^{T}\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{1:j-1}}\right)$ and $\vec{k}^{T}\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{j+1:n}}\right)$. $k\left(\overrightarrow{x_{n+1}},\overrightarrow{x_{n+1}}\right)$ is kernel function value between $\overrightarrow{x_{n+1}}$ and itself.

As we have known which sample of $X$ will be replaced by $\overrightarrow{x_{n+1}}$, the $j$ is a determined value.

Then we define some temporary variables.

$$\begin{aligned} \vec{r} &= \mathbf{K}_{new}(:,j)-\mathbf{K}_{old}(:,j) \\ \mathbf{A} &= \mathbf{K}_{old}^{-1}-\frac{\mathbf{K}_{old}^{-1}\cdot\vec{r}\cdot\mathbf{K}_{old}^{-1}(j,:)}{1+\vec{r}^{T}\cdot\mathbf{K}_{old}^{-1}(:,j)} \end{aligned} \tag{15}$$

Then the $\mathbf{K}_{new}^{-1}$ will be calculated as

$$\mathbf{K}_{new}^{-1}=\mathbf{A}-\frac{\mathbf{A}(:,j)\cdot\vec{r}^{T}\cdot\mathbf{A}}{1+\vec{r}^{T}\cdot\mathbf{A}(:,j)} \tag{16}$$

A demonstration is

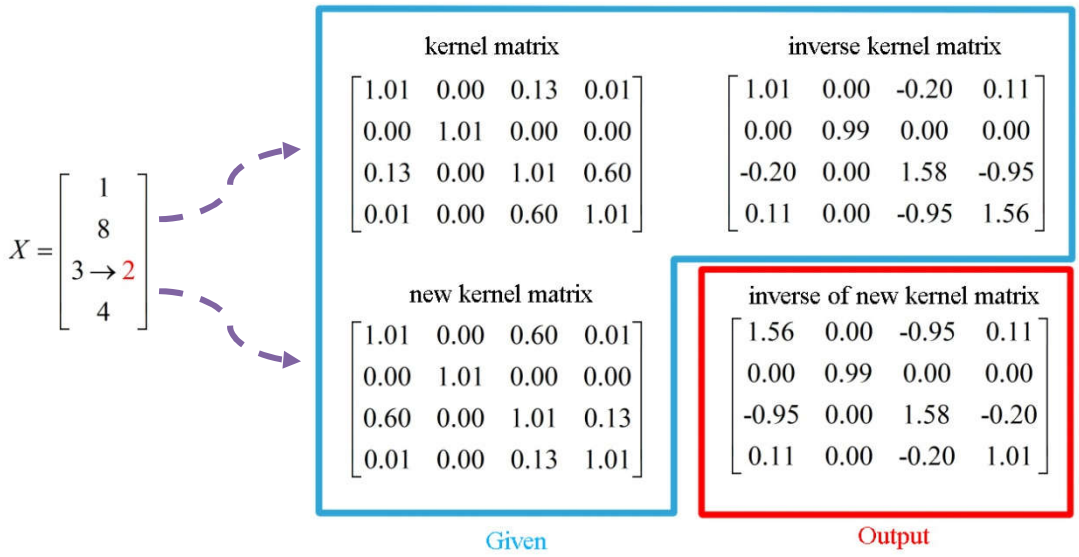


Figure.3 Demonstration of Sherman-Morrison Update

The kernel matrix and inverse of it in the upper part of figure is calculated from the "old" kernel as the first 4 elements of $X$. As the element "3" is replaced by the new sample "2". The matrix enclosed by blue is matrixes known or easy to be calculated, and the matrix enclosed by red is the result of (16) as the new kernel matrix $\mathbf{K}_{new}^{-1}$.

# Reference


[1] https://rosettacode.org/wiki/Cholesky_decomposition

[2] https://makarandtapaswi.wordpress.com/2011/07/08/cholesky-decomposition-for-matrix-inversion/

[3] Nguyen-Tuong, Duy, and Jan Peters. "Incremental online sparsification for model learning in real-time robot control." *Neurocomputing* 74.11 (2011): 1859-1867.

[4] "Kernel Adaptive Filtering", Jose C. Principe and Weifeng Liu, Computational Neuro Engineering Laboratory (CNEL), University of Florida

[5] https://en.wikipedia.org/wiki/Sherman%E2%80%93Morrison_formula