ML2R Theory Nuggets
Centering Data- and Kernel Matrices

Christian Bauckhage*
Machine Learning Rhine-Ruhr
Fraunhofer IAIS
St. Augustin, Germany

Pascal Welke†
Machine Learning Rhine-Ruhr
University of Bonn
Bonn, Germany

ABSTRACT

We discuss the notion of centered data matrices and show how to compute them using centering matrices. As centering matrices have many applications in data science and machine learning, we have a look at one such application and discuss how they allow for centering kernel matrices.

1 CENTERING DATA MATRICES

In data mining, machine learning, or pattern recognition, we often have to analyze or process a sample of n data points $\boldsymbol{x}_i \in \mathbb{R}^m$. This, in turn, frequently requires us to consider *zero mean* or *centered* data.

A simple canonical example of such a setting is the computation of the $m \times m$ (biased) sample covariance matrix

$$\boldsymbol{C} = \frac{1}{n} \sum_{i=1}^{n} \big[\boldsymbol{x}_i - \bar{\boldsymbol{x}}\big]\big[\boldsymbol{x}_i - \bar{\boldsymbol{x}}\big]^\top \tag{1}$$

where the vector

$$\bar{\boldsymbol{x}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \tag{2}$$

denotes the sample mean.

While this is a specific example, we note its following general aspect: The operation of subtracting the sample mean from each of the given data points

$$\boldsymbol{z}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}} \tag{3}$$

causes the resulting transformed data $\boldsymbol{z}_i$ to have the following mean

$$\begin{aligned} \bar{\boldsymbol{z}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{z}_i &= \frac{1}{n} \sum_{i=1}^{n} \big[\boldsymbol{x}_i - \bar{\boldsymbol{x}}\big] \\ &= \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i - \frac{1}{n} \sum_{i=1}^{n} \bar{\boldsymbol{x}} = \bar{\boldsymbol{x}} - \frac{1}{n} \cdot n \cdot \bar{\boldsymbol{x}} = \boldsymbol{0} \end{aligned} \tag{4}$$

In other words, the transformation in (3) causes the transformed data to be "of zero mean" or to be "centered at $\boldsymbol{0}$".

With respect to the sample covariance matrix in our introductory example, this means that we could also compute it as

$$\boldsymbol{C} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{z}_i \, \boldsymbol{z}_i^\top \tag{5}$$

* 0000-0001-6615-2128
† 0000-0002-2123-3781

What does any of this have to do with data matrices? Well, we may gather the n given data points $\boldsymbol{x}_i \in \mathbb{R}^m$ in an $m \times n$ data matrix

$$\boldsymbol{X} = \big[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\big] \tag{6}$$

and apply linear algebra to facilitate procedures such as the one we just went through. If we did this, we should hope that there is a linear algebraic mechanism to produce a centered data matrix

$$\boldsymbol{Z} = \big[\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n\big] \tag{7}$$

Indeed, linear algebra provides such a mechanism and we next discuss how it works.

In what follows, we let $\boldsymbol{1} \in \mathbb{R}^n$ denote the n-dimensional vector of all ones.

If n data points have been gathered in a data matrix $\boldsymbol{X}$, it is an easy exercise to verify that their sample mean in (2) can also be computed like this

$$\bar{\boldsymbol{x}} = \tfrac{1}{n} \boldsymbol{X}\boldsymbol{1} \tag{8}$$

When working with the vector of all ones, we can moreover consider the outer product $\bar{\boldsymbol{x}}\,\boldsymbol{1}^\top$ to obtain an $m \times n$ matrix

$$\bar{\boldsymbol{x}}\,\boldsymbol{1}^\top = \big[\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \ldots, \bar{\boldsymbol{x}}\big] \tag{9}$$

all of whose columns are copies of $\bar{\boldsymbol{x}}$.

This matrix now allows us to perform data centering, i.e. to subtract the sample mean from each of the given data points, by means of just a single matrix subtraction, namely

$$\boldsymbol{Z} = \boldsymbol{X} - \bar{\boldsymbol{x}}\,\boldsymbol{1}^\top \tag{10}$$

Given what we saw in (4), it is clear that the column mean of the resulting matrix $\boldsymbol{Z}$ will be $\boldsymbol{0}$. In other words, $\boldsymbol{Z}$ is a centered version of our original data matrix $\boldsymbol{X}$. Regarding our introductory example of the sample covariance matrix, this means that we can now compute is as

$$\boldsymbol{C} = \tfrac{1}{n} \boldsymbol{Z}\boldsymbol{Z}^\top \tag{11}$$

But let us keep going and develop even deeper insights into data centering! If we plug (8) into (10), we obtain the following crucial result

$$\boldsymbol{Z} = \boldsymbol{X} - \tfrac{1}{n} \boldsymbol{X}\boldsymbol{1}\boldsymbol{1}^\top = \boldsymbol{X}\big[\boldsymbol{I} - \tfrac{1}{n}\boldsymbol{1}\boldsymbol{1}^\top\big] \equiv \boldsymbol{X}\,\boldsymbol{J} \tag{12}$$

Here, $\boldsymbol{I}$ denotes the $n \times n$ identity matrix and the matrix

$$\boldsymbol{J} = \boldsymbol{I} - \tfrac{1}{n}\boldsymbol{1}\boldsymbol{1}^\top \tag{13}$$

is called a **centering matrix** because it centers the columns of matrix $\boldsymbol{X}$ at $\boldsymbol{0}$.

Centering matrices have a couple of important or convenient characteristics. For instance, it is easy to show that

(1) $\boldsymbol{J}$ is square, i.e. $\boldsymbol{J} \in \mathbb{R}^{n \times n}$
(2) $\boldsymbol{J}$ is symmetric, i.e. $\boldsymbol{J} = \boldsymbol{J}^\intercal$
(3) $\boldsymbol{J}$ is idempotent, i.e. $\boldsymbol{J}^k = \boldsymbol{J}$ for all $1 \leq k \in \mathbb{N}$
(4) $\boldsymbol{J}$ is positive semi-definite, i.e. $\boldsymbol{y}^\intercal \boldsymbol{J}\, \boldsymbol{y} \geq \boldsymbol{0}$ for all $\boldsymbol{y} \in \mathbb{R}^n$
(5) $\boldsymbol{J}$ is singular, i.e. *not* invertible, i.e. $\boldsymbol{J}^{-1}$ does not exists
(6) because of (4), all eigenvalues of $\boldsymbol{J}$ are non-negative
(7) because of (3), all eigenvalues of $\boldsymbol{J}$ are either 0 or 1
(8) because of (2) and (3), $\boldsymbol{J}$ is an orthogonal projection matrix
(9) the trace of $\boldsymbol{J}$ amounts to

$$\mathrm{tr}\big[\boldsymbol{J}\big] = n \cdot \big(1 - \tfrac{1}{n}\big) = n - 1$$

Regarding our introductory example of computing the sample covariance matrix, we can make use of some of the above properties to obtain

$$\boldsymbol{C} = \tfrac{1}{n}\, \boldsymbol{Z}\boldsymbol{Z}^\intercal \tag{14}$$

$$= \tfrac{1}{n}\, \big[\boldsymbol{X}\,\boldsymbol{J}\big]\big[\boldsymbol{X}\,\boldsymbol{J}\big]^\intercal \tag{15}$$

$$= \tfrac{1}{n}\, \boldsymbol{X}\,\boldsymbol{J}\boldsymbol{J}^\intercal \boldsymbol{X}^\intercal \tag{16}$$

$$= \tfrac{1}{n}\, \boldsymbol{X}\,\boldsymbol{J}\boldsymbol{J}\,\boldsymbol{X}^\intercal \tag{17}$$

$$= \tfrac{1}{n}\, \boldsymbol{X}\,\boldsymbol{J}\,\boldsymbol{X}^\intercal \tag{18}$$

Since this result may look curious at first sight, we will briefly verify it from a different point of view.

If we expand the outer products in the summation in (1), we obtain the following fairly well known decomposition of the sample covariance matrix

$$\boldsymbol{C} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\intercal - \frac{2}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \bar{\boldsymbol{x}}^\intercal + \frac{1}{n} \sum_{i=1}^{n} \bar{\boldsymbol{x}} \bar{\boldsymbol{x}}^\intercal \tag{19}$$

$$= \frac{1}{n}\, \boldsymbol{X}\boldsymbol{X}^\intercal - 2\, \bar{\boldsymbol{x}} \bar{\boldsymbol{x}}^\intercal + \bar{\boldsymbol{x}} \bar{\boldsymbol{x}}^\intercal \tag{20}$$

$$= \frac{1}{n}\, \boldsymbol{X}\boldsymbol{X}^\intercal - \bar{\boldsymbol{x}} \bar{\boldsymbol{x}}^\intercal \tag{21}$$

If we next plug (8) into this expression, we realize that the sample covariance matrix can also be written as

$$\boldsymbol{C} = \tfrac{1}{n}\, \boldsymbol{X}\boldsymbol{X}^\intercal - \tfrac{1}{n^2}\, \big[\boldsymbol{X}\boldsymbol{1}\big]\big[\boldsymbol{X}\boldsymbol{1}\big]^\intercal \tag{22}$$

$$= \tfrac{1}{n}\, \boldsymbol{X}\boldsymbol{X}^\intercal - \tfrac{1}{n^2}\, \boldsymbol{X}\,\boldsymbol{1}\boldsymbol{1}^\intercal \boldsymbol{X}^\intercal \tag{23}$$

$$= \tfrac{1}{n}\, \boldsymbol{X}\, \big[\boldsymbol{X}^\intercal - \tfrac{1}{n}\boldsymbol{1}\boldsymbol{1}^\intercal \boldsymbol{X}^\intercal\big] \tag{24}$$

$$= \tfrac{1}{n}\, \boldsymbol{X}\, \big[\boldsymbol{I} - \tfrac{1}{n}\boldsymbol{1}\boldsymbol{1}^\intercal\big]\, \boldsymbol{X}^\intercal \tag{25}$$

$$= \tfrac{1}{n}\, \boldsymbol{X}\,\boldsymbol{J}\,\boldsymbol{X}^\intercal \tag{26}$$

which does indeed corroborate the result in (18).

2 CENTERING KERNEL MATRICES

Many algorithms for data mining, machine learning, and pattern recognition work with Gram matrices

$$\boldsymbol{G} = \boldsymbol{X}^\intercal \boldsymbol{X} \tag{27}$$

which are matrices whose elements are given by $[\boldsymbol{G}]_{ij} = \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$ and thus only depend on inner products between given data points. This allows for invoking the **kernel trick** where we replace inner products by evaluations of a Mercer kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \big\langle \boldsymbol{\varphi}_i \,\big|\, \boldsymbol{\varphi}_j \big\rangle \tag{28}$$

Here, $\boldsymbol{\varphi}_i \equiv \boldsymbol{\varphi}(\boldsymbol{x}_i)$ where $\boldsymbol{\varphi} : \mathbb{R}^m \to \mathbb{H}$ is some generally unknown transformation from the data space into a potentially infinite dimensional feature space. The function $\langle \cdot \,|\, \cdot \rangle : \mathbb{H} \times \mathbb{H} \to \mathbb{R}$ denotes the inner product in that feature space.

To stress the whole point of our following discussion, we remark that it sounds formidable to work with unknown transformations into potentially infinite dimensional spaces. Recall, however, that we only do this *implicitly* when using *explicitly* given kernel functions. In other words, while it may be impossible to practically compute the inner product on the right of equation (28), it is usually possible to evaluate the kernel function on its left hand side.

With respect to Gram matrices, this is to say that invoking the kernel trick is to replace the $n \times n$ matrix $\boldsymbol{G}$ by an $n \times n$ kernel matrix $\boldsymbol{K}$ where

$$\big[\boldsymbol{K}\big]_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{29}$$

Conceptually, or from the point of view of "pen-and-paper math", this is the same as introducing a feature space data matrix

$$\boldsymbol{\Phi} = \Big[\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_n\Big] \tag{30}$$

and then replacing $\boldsymbol{G}$ by an $n \times n$ matrix $\boldsymbol{\Phi}^\intercal \boldsymbol{\Phi}$ where

$$\big[\boldsymbol{\Phi}^\intercal \boldsymbol{\Phi}\big]_{ij} = \big\langle \boldsymbol{\varphi}_i \,\big|\, \boldsymbol{\varphi}_j \big\rangle \tag{31}$$

Now, if we kernelize an algorithm that requires us to work with centered data $\boldsymbol{z}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}} \in \mathbb{R}^m$, we must make sure that their feature space representations $\boldsymbol{\psi}_i = \boldsymbol{\varphi}_i - \bar{\boldsymbol{\varphi}} \in \mathbb{H}$ are centered, too.

Again, this sounds formidable, because we just emphasized that it may be practically impossible to compute the underlying transformations. How are we then supposed to compute the feature space mean required for feature space centering? But let us do some "pen-and-paper math" to see what we are actually dealing with.

On a conceptual or symbolic level, we can easily work with the feature space data matrix $\boldsymbol{\Phi}$ in (30). Our insights form the previous section then tell us that

$$\boldsymbol{\Psi} = \boldsymbol{\Phi}\,\boldsymbol{J} = \boldsymbol{\Phi} - \tfrac{1}{n}\, \boldsymbol{\Phi}\boldsymbol{1}\boldsymbol{1}^\intercal \tag{32}$$

will be a centered matrix. However, if we cannot even write computer code to compute $\boldsymbol{\Phi}$, we certainly cannot write computer code to center it. Yet, what we are actually interested in when kernelizing an algorithm is to compute the Gramian $\boldsymbol{\Psi}^\intercal \boldsymbol{\Psi}$ and we will see next that this is practically feasible.

In fact, this is easy to see: Using the definition in (32) together with general properties of the centering matrix $\boldsymbol{J}$, we have

$$\boldsymbol{\Psi}^\top \boldsymbol{\Psi} = \big[\boldsymbol{\Phi}\,\boldsymbol{J}\big]^\top \big[\boldsymbol{\Phi}\,\boldsymbol{J}\big] \tag{33}$$

$$= \boldsymbol{J}^\top \boldsymbol{\Phi}^\top \boldsymbol{\Phi}\,\boldsymbol{J} \tag{34}$$

$$= \boldsymbol{J}\,\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\,\boldsymbol{J} \tag{35}$$

At this point, we note that it is actually possible to compute the entries of the feature space Gramian $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$, because

$$\big[\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\big]_{ij} = \big\langle \boldsymbol{\varphi}_i \,\big|\, \boldsymbol{\varphi}_j \big\rangle = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \big[\boldsymbol{K}\big]_{ij} \tag{36}$$

In other words, we may not be able to practically compute matrix $\boldsymbol{\Phi}$ but we are able to compute $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ because its entries can be obtained by evaluating a computable kernel function. All in all, we therefore have the crucial result that

$$\boldsymbol{\Psi}^\top \boldsymbol{\Psi} = \boldsymbol{J}\,\boldsymbol{K}\,\boldsymbol{J} \tag{37}$$

At this point, we have successfully shown that it is feasible to practically compute (and therefore work with) matrix $\boldsymbol{\Psi}^\top\boldsymbol{\Psi}$. **But let us once again keep going and develop deeper insights into what it means to center a kernel matrix.**

To begin with, we emphasize, that we have an equation that relates the kernel matrix $\boldsymbol{K}$ to the feature space data matrix $\boldsymbol{\Phi}$, namely

$$\boldsymbol{K} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi} \tag{38}$$

Given that we can compute $\boldsymbol{\Psi}^\top\boldsymbol{\Psi}$ for the centered feature space data matrix $\boldsymbol{\Psi}$, we now define

$$\boldsymbol{K}_c = \boldsymbol{\Psi}^\top\boldsymbol{\Psi} \tag{39}$$

and henceforth call $\boldsymbol{K}_c$ the *centered* kernel matrix.

Concerning the centered kernel matrix, we observe the following

$$\boldsymbol{K}_c = \boldsymbol{J}\,\boldsymbol{K}\,\boldsymbol{J} \tag{40}$$

$$= \big[\boldsymbol{I} - \tfrac{1}{n}\,\boldsymbol{1}\boldsymbol{1}^\top\big]\,\boldsymbol{K}\,\big[\boldsymbol{I} - \tfrac{1}{n}\,\boldsymbol{1}\boldsymbol{1}^\top\big] \tag{41}$$

$$= \big[\boldsymbol{K} - \tfrac{1}{n}\,\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{K}\big]\,\big[\boldsymbol{I} - \tfrac{1}{n}\,\boldsymbol{1}\boldsymbol{1}^\top\big] \tag{42}$$

$$= \boldsymbol{K} - \tfrac{1}{n}\,\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{K} - \tfrac{1}{n}\,\boldsymbol{K}\boldsymbol{1}\boldsymbol{1}^\top + \tfrac{1}{n^2}\,\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{K}\,\boldsymbol{1}\boldsymbol{1}^\top \tag{43}$$

While this looks interesting, we may again wonder if there is an intuition behind this expression?

To develop a better understanding of the nature of matrix $\boldsymbol{K}_c$, we next study its individual entries $[\boldsymbol{K}_c]_{ij}$.

Just as we can conceptually work with the feature space data matrix $\boldsymbol{\Phi}$, we can conceptually work with the feature space sample mean

$$\bar{\boldsymbol{\varphi}} = \frac{1}{n}\,\boldsymbol{\Phi}\,\boldsymbol{1} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\varphi}_i \tag{44}$$

Using this feature space sample mean, we can write the individual entries of $\boldsymbol{K}_c$ as

$$\big[\boldsymbol{K}_c\big]_{ij} = \big[\boldsymbol{\Psi}^\top\boldsymbol{\Psi}\big]_{ij} = \big\langle \boldsymbol{\psi}_i \,\big|\, \boldsymbol{\psi}_j \big\rangle = \big\langle \boldsymbol{\varphi}_i - \bar{\boldsymbol{\varphi}} \,\big|\, \boldsymbol{\varphi}_j - \bar{\boldsymbol{\varphi}} \big\rangle \tag{45}$$

And, if we expand these inner products between two centered feature space points, we obtain

$$\begin{aligned}\big\langle \boldsymbol{\varphi}_i - \bar{\boldsymbol{\varphi}} \,\big|\, \boldsymbol{\varphi}_j - \bar{\boldsymbol{\varphi}} \big\rangle = \big\langle \boldsymbol{\varphi}_i \,\big|\, \boldsymbol{\varphi}_j \big\rangle &- \big\langle \boldsymbol{\varphi}_i \,\big|\, \bar{\boldsymbol{\varphi}} \big\rangle \\ &- \big\langle \bar{\boldsymbol{\varphi}} \,\big|\, \boldsymbol{\varphi}_j \big\rangle + \big\langle \bar{\boldsymbol{\varphi}} \,\big|\, \bar{\boldsymbol{\varphi}} \big\rangle\end{aligned} \tag{46}$$

Using the definition of the feature space sample mean in (44) and the bilinearity of the inner product, this can also be written as

$$\begin{aligned}\big\langle \boldsymbol{\varphi}_i - \bar{\boldsymbol{\varphi}} \,\big|\, \boldsymbol{\varphi}_j - \bar{\boldsymbol{\varphi}} \big\rangle = \big\langle \boldsymbol{\varphi}_i \,\big|\, \boldsymbol{\varphi}_j \big\rangle &\\ - \tfrac{1}{n}\sum_{j=1}^{n}\big\langle \boldsymbol{\varphi}_i \,\big|\, \boldsymbol{\varphi}_j \big\rangle &\\ - \tfrac{1}{n}\sum_{i=1}^{n}\big\langle \boldsymbol{\varphi}_i \,\big|\, \boldsymbol{\varphi}_j \big\rangle &\\ + \tfrac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\big\langle \boldsymbol{\varphi}_i \,\big|\, \boldsymbol{\varphi}_j \big\rangle &\end{aligned} \tag{47}$$

Given this expansion, we can now resort to (28) and write all the inner products on the right hand side of (47) in terms of kernel functions

$$\begin{aligned}\big\langle \boldsymbol{\varphi}_i - \bar{\boldsymbol{\varphi}} \,\big|\, \boldsymbol{\varphi}_j - \bar{\boldsymbol{\varphi}} \big\rangle = K(\boldsymbol{x}_i, \boldsymbol{x}_j) &\\ - \tfrac{1}{n}\sum_{j=1}^{n}K(\boldsymbol{x}_i, \boldsymbol{x}_j) &\\ - \tfrac{1}{n}\sum_{i=1}^{n}K(\boldsymbol{x}_i, \boldsymbol{x}_j) &\\ + \tfrac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}K(\boldsymbol{x}_i, \boldsymbol{x}_j) &\end{aligned} \tag{48}$$

At this point, we observe that all terms on the right of (48) can be expressed in terms of our original kernel matrix $\boldsymbol{K}$. In particular, we have

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{e}_i^\top\,\boldsymbol{K}\,\boldsymbol{e}_j \tag{49}$$

$$\sum_{j=1}^{n}K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{e}_i^\top\,\boldsymbol{K}\,\boldsymbol{1} \tag{50}$$

$$\sum_{i=1}^{n}K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{1}^\top\,\boldsymbol{K}\,\boldsymbol{e}_j \tag{51}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{1}^\top\,\boldsymbol{K}\,\boldsymbol{1} \tag{52}$$

where $\boldsymbol{e}_i, \boldsymbol{e}_j \in \mathbb{R}^n$ denote the i-th and j-th standard basis vector. But this is to say that we can express the (i, j) entry of the centered kernel matrix as

$$\big[\boldsymbol{K}_c\big]_{ij} = \boldsymbol{e}_i^\top\boldsymbol{K}\boldsymbol{e}_j - \tfrac{1}{n}\boldsymbol{e}_i^\top\boldsymbol{K}\boldsymbol{1} - \tfrac{1}{n}\boldsymbol{1}^\top\boldsymbol{K}\boldsymbol{e}_j + \tfrac{1}{n^2}\boldsymbol{1}^\top\boldsymbol{K}\boldsymbol{1} \tag{53}$$

Since this is beginning to resemble the expression in (43), we further note the following: The expression in (49) is but the (i, j) entry of $\boldsymbol{K}$. The expression (50) is the i-th component of a column vector, namely $\boldsymbol{K}\,\boldsymbol{1}$. The expression in (51) is the j-th component of a row vector, namely $\boldsymbol{1}^\top\boldsymbol{K}$. And $\boldsymbol{1}^\top\boldsymbol{K}\,\boldsymbol{1}$ is a scalar. Turning these latter objects into $n \times n$ matrices can be accomplished like this: $\boldsymbol{K}\,\boldsymbol{1}\boldsymbol{1}^\top$, $\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{K}$, and $\boldsymbol{1}^\top\boldsymbol{K}\,\boldsymbol{1}\boldsymbol{1}\boldsymbol{1}^\top = \boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{K}\,\boldsymbol{1}\boldsymbol{1}^\top$. Using these matrices, we therefore have

$$\boldsymbol{K}_c = \boldsymbol{K} - \tfrac{1}{n}\,\boldsymbol{K}\,\boldsymbol{1}\boldsymbol{1}^\top - \tfrac{1}{n}\,\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{K} + \tfrac{1}{n^2}\,\boldsymbol{1}\boldsymbol{1}^\top\boldsymbol{K}\,\boldsymbol{1}\boldsymbol{1}^\top \tag{54}$$

which corresponds exactly to the result we found in equation (43).

3 SUMMARY OF MAIN RESULTS

In data mining, pattern recognition, and machine learning, we often work with data matrices

$$\boldsymbol{X} = \big[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\big]$$

whose n columns vectors $\boldsymbol{x}_i$ have a mean of $\bar{\boldsymbol{x}}$. For many tasks, however, we need to turn such data matrices into centered data matrices

$$\boldsymbol{Z} = \big[\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n\big]$$

whose n columns vectors $\boldsymbol{z}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}$ have a mean of $\boldsymbol{0}$. Considering the centering matrix

$$\boldsymbol{J} = \boldsymbol{I} - \tfrac{1}{n}\,\boldsymbol{1}\boldsymbol{1}^\top$$

we can compute the centered version $\boldsymbol{Z}$ of data matrix $\boldsymbol{X}$ as follows

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{J}$$

The "outer product" of a centered matrix $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{J}$ with itself can be computed in terms of the original matrix $\boldsymbol{X}$, namely

$$\begin{aligned}\boldsymbol{Z}\boldsymbol{Z}^\top &= \big[\boldsymbol{X}\,\boldsymbol{J}\big]\big[\boldsymbol{X}\,\boldsymbol{J}\big]^\top \\ &= \boldsymbol{X}\,\boldsymbol{J}\boldsymbol{J}^\top\boldsymbol{X}^\top \\ &= \boldsymbol{X}\,\boldsymbol{J}\boldsymbol{J}\,\boldsymbol{X}^\top \\ &= \boldsymbol{X}\,\boldsymbol{J}\,\boldsymbol{X}^\top\end{aligned}$$

This uses the fact that the centering matrix $\boldsymbol{J}$ is symmetric and idempotent.

The "inner product" of a centered matrix $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{J}$ with itself can also be computed in terms of the original matrix $\boldsymbol{X}$, namely

$$\begin{aligned}\boldsymbol{Z}^\top\boldsymbol{Z} &= \big[\boldsymbol{X}\,\boldsymbol{J}\big]^\top\big[\boldsymbol{X}\,\boldsymbol{J}\big] \\ &= \boldsymbol{J}^\top\boldsymbol{X}^\top\boldsymbol{X}\,\boldsymbol{J} \\ &= \boldsymbol{J}\,\boldsymbol{X}^\top\boldsymbol{X}\,\boldsymbol{J}\end{aligned}$$

This again uses the symmetry of the centering matrix $\boldsymbol{J}$.

When working with algorithms that involve computations with Gram matrices $\boldsymbol{X}^\top\boldsymbol{X}$, we can invoke the kernel trick and replace the Gramian by a kernel matrix $\boldsymbol{K}$. The corresponding centered kernel matrix is given by

$$\boldsymbol{K}_c = \boldsymbol{J}\,\boldsymbol{K}\,\boldsymbol{J}$$

4 A SHORT NOTE ON IMPLEMENTATION

As it is customary for this series to provide simple *NumPy* implementations of the proposed methods, we will do just that. A centering matrix for n data points can be obtained with

```
import numpy as np

def get_centering_matrix(n):
    return np.eye(n) - 1. / n * np.ones([n,1])
```

While this seems like a straightforward implementation of (13) at first glance, we note one slight difference: *NumPy* allows for so called *broadcasting* of the *vector* `np.ones([1,n])` implementing $\boldsymbol{1}$ to `np.eye(n)` implementing the $n \times n$ matrix $\boldsymbol{I}$ if the dimensions are compatible. This allows to implement (13) without explicitly creating an $n \times n$ matrix filled with ones, which becomes very useful for larger problem instances.

5 OUTLOOK TO APPLICATION EXAMPLES

As we saw in our introductory example, centered data matrices play a crucial role in computing sample covariance matrices. They are thus important for methods such as PCA [7] or CCA [10] which build on top of covariance matrices. But centered data matrices occur in other contexts, too. For instance, they may form the basis of simple yet efficient clustering algorithms [1, 2].

Centered Gram- or kernel matrices play crucial roles in multidimensional scaling [8] or kernel PCA [9]. Especially the latter allows for interesting approaches to text analysis [3–6] which we discuss in our ML2R Coding Nuggets series.

ACKNOWLEDGMENTS

This material was produced within the Competence Center for Machine Learning Rhine-Ruhr (**ML2R**) which is funded by the Federal Ministry of Education and Research of Germany (grant no. 01IS18038C). The authors gratefully acknowledge this support.

REFERENCES

[1] C. Bauckhage, E. Brito, K. Cvejoski, C. Ojeda, R. Sifa, and S. Wrobel. 2017. Ising Models for Binary Clustering via Adiabatic Quantum Computing. In *Proc. EMMCVPR*. Springer.
[2] C. Bauckhage, C. Ojeda, R. Sifa, and S. Wrobel. 2018. Adiabatic Quantum Computing for Kernel k=2 Means Clustering. In *Proc. KDML-LWDA*.
[3] M. Beeksma, M. van Gompel, F. Kunneman, L. Onrust, B. Regnerus, D. Vinke, E. Brito, C. Bauckhage, and R. Sifa. 2018. Detecting and Correcting Spelling Errors in High-quality Dutch Wikipedia Text. *Computational Linguistics in the Netherlands J.* 8 (2018), 122–137.
[4] E. Brito, B. Georgiev, D. Domingo-Fernandez, C.T. Hoyt, and C. Bauckhage. 2019. RatVec: A General Approach for Low-dimensional Distributed Vector Representations via Rational Kernels. In *Proc. KDML-LWDA*.
[5] E. Brito, R. Sifa, and C. Bauckhage. 2017. KPCA Embeddings: An Unsupervised Approach to Learn Vector Representations of Finite Domain Sequences. In *Proc. KDML-LWDA*.
[6] V. Gupta, S. Giesselbach, S. Rüping, and C. Bauckhage. 2019. Improving Word Embeddings Using Kernel PCA. In *Proc. Workshop on Representation Learning for NLP @ ACL*.
[7] I.T. Jolliffe. 1986. *Principal Component Analysis*. Springer.
[8] J.B. Kruskal and M. Wish. 1978. *Multidimensional Scaling*. SAGE Publications.
[9] B. Schölkopf, A. Smola, and K.-R. Müller. 1998. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation* 10 (1998).
[10] B. Thompson. 1985. *Canonical Correlation Analysis*. SAGE Publications.