

Change-Point Detection using Krylov Subspace Learning

Tsuyoshi Idé^{*} and Koji Tsuda[†]

Abstract

We propose an efficient algorithm for principal component analysis (PCA) that is applicable when only the inner product with a given vector is needed. We show that Krylov subspace learning works well both in matrix compression and implicit calculation of the inner product by taking full advantage of the arbitrariness of the seed vector. We apply our algorithm to a PCA-based change-point detection algorithm, and show that it results in about 50 times improvement in computational time.

Keywords: PCA, Krylov subspace, inner product, change-point detection

1 Introduction

Principal component analysis (PCA) is widely used feature extraction technique in data mining. Recently, PCA has been found to be useful also in the change-point (CP) detection task for real-valued time-series data. In the PCA-based CP detection algorithm named singular-spectrum transformation (SST [8, 6]), PCA is performed at each point of time on subsequences taken from both the past and present domains (see Fig. 1). The principal components are then compared to compute the degree of change, or the CP score.

Since SST does not employ any specific generative models, it is relatively robust and flexible against heterogeneities of the input signal. In spite of this useful feature, however, SST has a difficulty for practical use: The computational cost of singular value decomposition (SVD) is too high to be repeated over all of the time points.

To speed up PCA (or SVD), there have been a number of attempts so far. Recent studies include sampling-based techniques [11, 4] and online SVD algorithms [12]. However, these techniques are not useful in SST, since the matrix in SST is generally *dense*, and the number of subsequences is too small to use sampling-based techniques. Also, most of the online SVD algorithms implicitly assume the continuity with the past, which is not

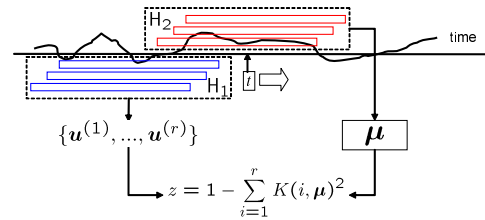


Figure 1: Overview of SST.

acceptable in the CP detection task.

In SST, the CP score is computed based on the inner product

$$(1.1) \quad K(i, \boldsymbol{\mu}) \equiv \boldsymbol{\mu}^\top \mathbf{u}^{(i)},$$

where $\boldsymbol{\mu}$ is the highest principal component of the present state, and the $\mathbf{u}^{(i)}$ s are principal components in the past, as shown in Fig. 1. In later sections, we will show that K can be calculated without performing SVD for the $\mathbf{u}^{(i)}$ s, based on our implicit Krylov approximation.

In machine learning, the Krylov subspace method has mainly been used as a black-box numerical solvers in the form of the conjugate gradient and the Lanczos methods [5], which are known to work quite well in *sparse* systems. Few attempts have been made to incorporate the concept of Krylov subspace into machine learning algorithms, except for a very recent study [3], where the Gaussian transform is simply combined to speed up vector-matrix multiplications. The essential idea of the present work is to take full advantage of the “seed” vector of the Krylov subspace. To the best of the authors’ knowledge, this is the first work which shows that the Krylov subspace learning enables us to do an implicit inner product calculation via an appropriate choice of the seed vector. Using real-world data, we will show that our new method makes the SST about 50 times faster.

Layout of this paper is as follows: Section 2 briefly reviews the SST. Our implicit Krylov approximation is introduced in Section 3, and is incorporated into SST in Section 4. Experimental results are given in Section 5. Section 6 concludes the paper.

^{*}IBM Research, Tokyo Research Laboratory. E-mail: good-idea@jp.ibm.com

[†]Max Planck Institute for Biological Cybernetics, koji.tsuda@tuebingen.mpg.de

2 Change-point detection algorithm

For the time-series data $\{s_t \in \mathbb{R} \mid t = 1, 2, \dots\}$, define a subsequence of length w as

$$\mathbf{s}(t) \equiv (s_{t-w+1}, \dots, s_{t-1}, s_t)^\top \in \mathbb{R}^w$$

where the superscript \top represents transpose. We assume that the data points are collected at constant intervals. At each time t , let \mathbf{H}_1 and \mathbf{H}_2 be matrices containing n subsequences defined as

$$\begin{aligned} \mathbf{H}_1(t) &\equiv [\mathbf{s}(t-n), \dots, \mathbf{s}(t-2), \mathbf{s}(t-1)] \\ \mathbf{H}_2(t) &\equiv [\mathbf{s}(t-n+\gamma), \dots, \mathbf{s}(t-1+\gamma)], \end{aligned}$$

where γ is a positive integer. Fig. 1 shows an example where three subsequences are taken both in the vicinity of the present time and in the past.

The column space of $\mathbf{H}_1(t)$, the space spanned by the column vectors, should contain the information about the patterns appearing on the past domain of the time series. The SST utilizes the principal components as typical representative patterns of the column space: Find the r ($< w, n$) top left singular vectors of $\mathbf{H}_1(t)$, $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(r)}$. We assume these are orthonormal. Hereafter, we omit the argument t unless confusion is likely. Let the subspace spanned by these vectors be

$$(2.2) \quad \mathcal{H}_r \equiv \text{span}\{\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots, \mathbf{u}^{(r)}\}.$$

Similarly, we can get the representative patterns around the present time t by performing the SVD of \mathbf{H}_2 . We use the top principal component $\boldsymbol{\mu}$ of \mathbf{H}_2 as the representative pattern (for extension to include more principal components, see Appendix A).

Unlike [6], we define the CP score z at time t as

$$(2.3) \quad z \equiv 1 - \sum_{i=1}^r K(i, \boldsymbol{\mu})^2,$$

which can be interpreted as the distance between the subspaces. Appendix A explains the detail.

Since it is empirically true that the score is not very sensitive to the choices of n and γ [6], we set $n = w$ and $\gamma = w/2$. For w , a value less than 100 typically works. An appropriate preprocess (e.g. down-sampling) can be used to adjust w to this range. Empirically, a value of three or four works well for r even when w is on the order of 100.

3 Implicit Krylov approximation

In this section, we introduce a matrix compression algorithm to compute $K(i, \boldsymbol{\mu})$. By definition, the singular vectors $\mathbf{u}^{(i)}$ are in the column space of \mathbf{H}_1 . Instead of using the full column space, we attempt

to use a k -dimensional subspace \mathcal{V}_k , and to reduce the original eigen problem to a $k \times k$ matrix problem (assuming $r < k < w$). Notice that what we want is not singular vectors themselves but the inner product w.r.t. a given vector $\boldsymbol{\mu}$. Thus we take full advantage of $\boldsymbol{\mu}$ for constructing \mathcal{V}_k , as suggested in Fig. 2. Details are explained below.

3.1 Implicit calculation of K . Let $\mathbf{q}_1, \dots, \mathbf{q}_k$ be orthonormal bases of \mathcal{V}_k . Suppose that $\mathbf{q}_1 = \boldsymbol{\mu}$. The $\mathbf{u}^{(i)}$ s were originally defined as the eigenvectors of $\mathbf{C} \equiv \mathbf{H}_1 \mathbf{H}_1^\top$. In \mathcal{V}_k , the eigen equation formally reads

$$(3.4) \quad \mathbf{Q}_k^\top \mathbf{C} \mathbf{Q}_k \mathbf{x} = \lambda \mathbf{x},$$

where $\mathbf{Q}_k \equiv [\mathbf{q}_1, \dots, \mathbf{q}_k]$. If $k = w$, we see that this eigen equation is exactly the same as the original problem, and $\mathbf{Q}_k \mathbf{x}$ corresponds to the original eigenvector. If $k < w$, on the other hand, this equation will give an approximated solution.

Although we have not yet specified \mathbf{q}_α s except for \mathbf{q}_1 , imagine that we have found the r top eigenvectors $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(r)} \in \mathbb{R}^k$ somehow by solving Eq. (3.4). Then a $\mathbf{u}^{(i)}$ is approximately given by

$$(3.5) \quad \mathbf{u}^{(i)} \simeq \sum_{\alpha=1}^k x_\alpha^{(i)} \mathbf{q}_\alpha,$$

where $x_\alpha^{(i)}$ is the α -th element of $\mathbf{x}^{(i)}$. Then, thanks to the orthogonality of the \mathbf{q}_α s and the fact that $\mathbf{q}_1 = \boldsymbol{\mu}$, computing $\boldsymbol{\mu}^\top \mathbf{u}^{(i)}$ reduces to taking the first element of the $\mathbf{x}^{(i)}$ s. Explicitly, the kernel function is approximately given by

$$(3.6) \quad K(i, \boldsymbol{\mu}) \simeq x_1^{(i)},$$

which means that *the inner product can be computed directly from $\mathbf{x}^{(i)}$ s without explicitly using $\mathbf{u}^{(i)}$ s*.

Now our problems are: How to find \mathcal{V}_k so that the overlap between \mathcal{H}_r and \mathcal{V}_k is as large as possible, and how to efficiently find the eigenvectors of $\mathbf{Q}_k^\top \mathbf{C} \mathbf{Q}_k$. These problems will be discussed in the next subsections.

3.2 Krylov subspace. To answer the first question, let us consider the following problem:

Given an s -dimensional subspace $\mathcal{V}_s \subset \mathbb{R}^w$, construct a subspace \mathcal{V}_{s+1} by adding a vector to \mathcal{V}_s so that the increase of the overlap between \mathcal{V}_{s+1} and $\{\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(s)}\}$ is maximized.

Let us start with \mathcal{V}_1 spanned by $\boldsymbol{\mu}$. Recall that solving the eigen equation for \mathbf{C} is equivalent to the maximization problem of the Rayleigh quotient [5], which is given by

$$R(\mathbf{u}) = (\mathbf{u}^\top \mathbf{C} \mathbf{u}) / (\mathbf{u}^\top \mathbf{u}).$$

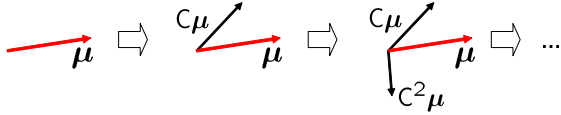


Figure 2: Krylov subspace construction with the seed vector μ .

To satisfy the requirement, when we construct $\mathcal{V}_2 = \text{span}\{\mu, \Delta\}$ by adding $\Delta \in \mathbb{R}^w$, the added vector should contain the steepest ascent direction of R given by

$$\left. \frac{d}{d\mathbf{u}} R(\mathbf{u}) \right|_{\mathbf{u}=\mu} = \frac{-2}{\mu^\top \mu} [R(\mu)\mu - C\mu].$$

Thus, if we choose $C\mu$ as Δ , $\text{span}\{\mu, C\mu\}$ contains this steepest direction.

Continuing this procedure, we see that a k -dimensional space

$$\mathcal{V}_k(\mu, C) \equiv \text{span}\{\mu, C\mu, \dots, C^{k-1}\mu\}$$

is the best k -dimensional subspace in terms of maximization of R , given μ . In other words, there are many choices¹ of a k -dimensional subspace over the entire column space of H_1 , among all of the choices, the subspace which has the largest weight of $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(r)}$ is $\mathcal{V}_k(\mu, C)$, under the constraint that μ is the starting base.

In mathematics, $\mathcal{V}_k(\mu, C)$ is called the *Krylov subspace* induced by μ and C [5]. Alternatively, one may say that μ is the *seed* of the Krylov subspace.

Figure 2 illustrates the construction procedure for $\mathcal{V}_k(\mu, C)$. Since $\mathcal{V}_k(\mu, C)$ is a subspace that condenses the information we want as densely as possible, reducing the problem within this space should be a good approximation.

3.3 Finding the orthonormal bases. To reduce the original eigen problem of C to the smaller problem defined in $\mathcal{V}_k(\mu, C)$, it seems that we would need to find $\mathbf{q}_1, \dots, \mathbf{q}_k$ explicitly (as will be shown, not necessary in fact). For that purpose, we can use the Gram-Schmidt orthogonalization or, equivalently, the (thin-) QR factorization of the Krylov matrix defined as

$$V_k(\mu, C) \equiv [\mu, C\mu, \dots, C^{k-1}\mu].$$

Due to the nature of these methods, the constraint $\mathbf{q}_1 = \mu$ is automatically satisfied. In addition and fortuitously, in the QR factorization of the Krylov matrix, a special and helpful property holds:

¹For example, a matrix compression technique in [2] produces another subspace, which is suboptimal in terms of the overlap with \mathcal{H}_r .

THEOREM 1. *The orthogonal matrix $Q_w \in \mathbb{R}^{w \times w}$ given by the QR factorization of $V_w(\mu, C)$ tridiagonalizes C .*

Proof (outline). Suppose that we have an orthogonal matrix Q_w which tridiagonalize C . Explicitly, $T_w \equiv Q_w^\top C Q_w$ is tridiagonal. Since $Q_w Q_w^\top$ is the identity matrix, and $Q_w^\top \mu = \mathbf{e}_1$, where $\mathbf{e}_1 \in \mathbb{R}^w$ is a unit vector corresponding to the first base, we see that

$$Q_w^\top V_k = [\mathbf{e}_1, T_w \mathbf{e}_1, \dots, T_w^{k-1} \mathbf{e}_1],$$

which is an upper triangular matrix. Thus, if we have a matrix which makes C tridiagonal, it must be a Q-factor of the QR factorization of V_r . Since the QR factorization is essentially unique [5], the Q-factor must tridiagonalize C . \square

Therefore, if $\mathbf{q}_1, \dots, \mathbf{q}_k$ are to be computed via the QR-factorization of $V_k(\mu, C)$, then Eq. (3.4) is an eigen equation for a tridiagonal matrix. Let $\alpha_1, \dots, \alpha_w$ and $\beta_1, \dots, \beta_{w-1}$ be the diagonal and subdiagonal elements of $T_w \equiv Q_w^\top C Q_w$. If we consider the s -th column of the equation $C_w Q_w = Q_w T_w$, it follows that

$$C\mathbf{q}_s = \alpha_s \mathbf{q}_s + \beta_{s-1} \mathbf{q}_{s-1} + \beta_s \mathbf{q}_{s+1},$$

where \mathbf{q}_s is the s -th column vector of Q_w . Using the orthogonal relation $\mathbf{q}_i^\top \mathbf{q}_j = \delta_{i,j}$, we immediately have $\alpha_s = \mathbf{q}_s^\top C \mathbf{q}_s$. In this way, it is easy to construct an algorithm to find α_s and β_s sequentially from this recurrent equation:

SUBROUTINE 1. Lanczos(C, μ, k) *Input* $C \in \mathbb{R}^{w \times w}$, $\mu \in \mathbb{R}^w$, and a positive integer $k (< w)$. *Initialize as* $\mathbf{r}_0 = \mu$, $\beta_0 = 1$, $\mathbf{q}_0 = 0$, and $s = 0$. *Repeat*
 $\mathbf{q}_{s+1} = \mathbf{r}_s / \beta_s$
 $s \leftarrow s + 1$
 $\alpha_s = \mathbf{q}_s^\top C \mathbf{q}_s$
 $\mathbf{r}_s = C \mathbf{q}_s - \alpha_s \mathbf{q}_s - \beta_{s-1} \mathbf{q}_{s-1}$
 $\beta_s = \|\mathbf{r}_s\|$
until $s = k$. *Return* $\{\alpha_1, \dots, \alpha_k\}$ and $\{\beta_1, \dots, \beta_{k-1}\}$.

By running this procedure up to $k < w$, we obtain $T_k (= Q_k^\top C Q_k)$ directly. Notice that we do *not* need to explicitly compute $\mathbf{q}_1, \dots, \mathbf{q}_k$. This tridiagonalization procedure is called the Lanczos algorithm.

3.4 Diagonalizing T_k . Now the first problem described at the end of Subsection 3.1 has been answered. Regarding the second problem, which is how to find the eigenvectors of the tridiagonal matrix T_k , fortunately there are extremely fast and stable algorithms. One of the best ways is to use the QL iteration (see, e.g. the `tqli` routine in [9]), which preserves the tridiagonal structure throughout the entire iteration process.

Notice that we do not need to explicitly compute either the $\mathbf{u}^{(i)}$ s or the inner product to get K . We call

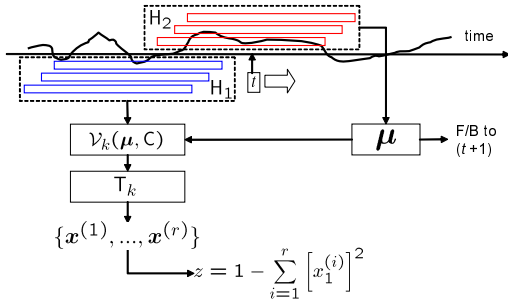


Figure 3: Overview of IKA-based SST.

this approach the *implicit Krylov approximation* (IKA) hereafter.

4 IKA-based change-point detection

4.1 Algorithm summary. The IKA needs μ as input (the top singular vector of H_2). Since the largest singular value does not have multiplicity ², it can be computed very efficiently using iterative methods such as the power method [5].

We summarized the IKA-based SST method in Fig. 3. At each t , we first compute μ . Then we run `Lanczos(C, μ, k)` to have T_k . Based on the r top eigenvectors $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(r)}$ of T_k , the CP score is given by $z \simeq 1 - \sum_{i=1}^r x_1^{(i)2}$. In the Figure, we also put the idea of feedback (to utilize the result at $t - 1$ for the initial vector at t). While the feedback speeds up the algorithm to some extent, the improvement is much smaller than that by the IKA.

For the dimension of the Krylov subspace $\mathcal{V}_k(\mu, C)$, one reasonable choice is

$$(4.7) \quad k = \begin{cases} 2r & r \in \text{even} \\ 2r - 1 & r \in \text{odd} \end{cases} .$$

The rationale of this rule is that the Krylov subspace is also the best one for the smallest eigenstates as well as for the largest [5], so k should be about twice r . Note that the IKA is independent of the choice of the SST-native parameters n and γ .

4.2 Remarks. The Lanczos algorithm is known to be numerically unstable in nature. In fact, if we explicitly compute the $\mathbf{u}^{(i)}$ s using Eq. (3.5), the resulting $\mathbf{u}^{(i)}$ s would suffer from pathological phenomena such as pseudo-degeneracies, especially in dense matrices (see, e.g., § 9.2 of [5]). In that case, we will have unreliable \mathcal{H}_r , resulting in an erroneously fluctuating z . The IKA

²We assume the input signal has been preprocessed so that it takes positive values in the majority of observations.

Table 1: Tested methods.

#	symbol	μ	feedback	$\{\mathbf{u}^{(i)}\}$	kernel
1	OI	power	no	OI	explicit
2	EM	EMPCA	no	EMPCA	explicit
3	OLF	power	yes	OI	explicit
4	EM.FB	EMPCA	yes	EMPCA	explicit
5	IKA	power	yes	-	implicit

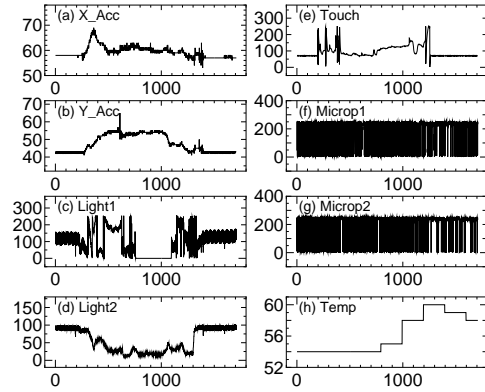


Figure 4: The phone1 data.

is carefully designed to avoid such difficulties which are very likely to occur in dense matrices.

5 Experiment

We implemented five different types of SST algorithms in Java as shown in Table 1. The first four explicitly compute the singular vectors using different routines: **power** (the power method), **OI** (orthogonal iteration [5]), and **EMPCA** (EM-PCA algorithm [10]). These were compared to our IKA-based SST algorithm. All calculations were done in a Java 1.4.2 virtual machine on a modest workstation (Pentium 4, 2.0 GHz, 1 GB memory). In the iterative algorithms, the convergence threshold was set to be 10^{-5} for the norm of the residual vectors.

The data used was the *phone1* data (Fig. 4) containing eight time series of various types measured by embedded sensors in a mobile phone [1, 7]. Each of the variables consists of 1,708 data points, but information about the sampling rate is not given. From the title attached to the data file, it seems that the data represents the actions of picking up the phone and laying it down.

5.1 Computational cost. We measured the computational times of these five SST algorithms. As a pre-process, the original signals were scaled to have unit variance and a mean of three. We imposed a periodic boundary condition on the data in performing SST. This

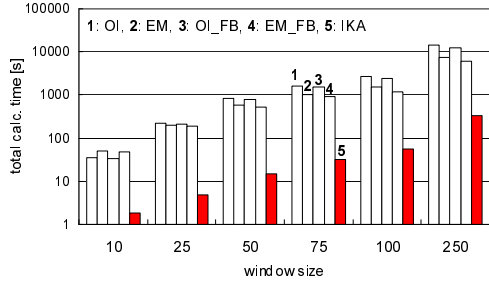


Figure 5: Total computation time of SST.

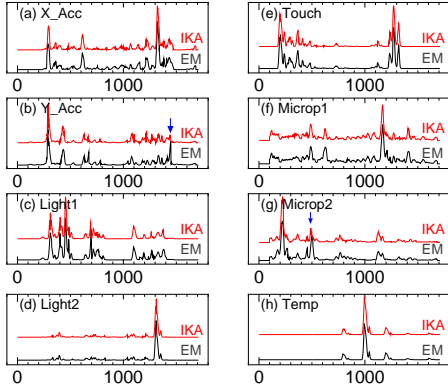


Figure 6: CP score of the phone1 data ($w = 50, r = 3$).

is to keep the number of data points the same over different w s. We used $(r, k) = (3, 5)$.

Figure 5 compares the computational times of the different algorithms on a logarithmic scale, averaged over five trials. We see that the improvement by the IKA-SST is drastic. It is about 50 times faster than the conventional SST methods for each w .

5.2 Numerical Errors. Notice that this was accomplished with no substantial approximation error. To see this, Fig. 6 compares the CP scores between EM and IKA for $w = 50$. As shown, the overall fit between the EM and IKA results is very good, although there are a few peaks which are not reproduced by IKA as indicated in Figs. 6 (b) and (g). Again, it is surprising that the IKA almost perfectly reproduces the results of EM, since IKA solves only 5×5 problems while EM performs the complete SVD for 50×50 matrices.

6 Concluding remarks

We have proposed a new PCA algorithm named the implicit Krylov approximation that is applicable when one is interested only in the inner product with a given vector. We showed that Krylov subspace learning ac-

complishes both matrix compression and implicit inner product calculation at the same time. We applied the IKA to the change-point detection task, and demonstrated that the IKA made the conventional SST algorithm about 50 times faster. Application of the IKA to other areas would be interesting future work.

A Appendix: Distance between subspaces

In this Appendix, we explain why the definition Eq. (2.3) is consistent.

A.1 Projection operators. In SST, the CP score is generally defined as the distance between the feature spaces in the past and present domains. Defining this distance is nontrivial because the dimensions of the subspaces can be different. Let \mathcal{S}_h be the feature space in the present domain. As Eq. (2.2),

$$\mathcal{S}_h \equiv \text{span}\{\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \dots, \boldsymbol{\mu}^{(h)}\},$$

where $\boldsymbol{\mu}^{(i)}$ s are the top h singular vectors of \mathbf{H}_2 . Notice that we are extending the theory to include $h > 1$.

Let \mathbf{P}_1 and \mathbf{P}_2 be the projection operators onto \mathcal{H}_r and \mathcal{S}_h , respectively. Using the singular vectors, these can be written as

$$(A.8) \quad \mathbf{P}_1 = \sum_{i=1}^r \mathbf{u}^{(i)} \mathbf{u}^{(i)\top} \quad \text{and} \quad \mathbf{P}_2 = \sum_{i=1}^h \boldsymbol{\mu}^{(i)} \boldsymbol{\mu}^{(i)\top}.$$

Let $\mathbf{x} \in \mathbb{R}^w$ be a vector in \mathcal{H}_r . Since similar subspaces should yield similar projections, it is natural to focus on comparing $\mathbf{P}_1 \mathbf{x}$ ($= \mathbf{x}$) and $\mathbf{P}_2 \mathbf{x}$. Clearly, if \mathcal{S}_h is perpendicular to \mathcal{H}_r , $\mathbf{P}_2 \mathbf{x}$ vanishes (see Fig. 7 (a)). In contrast, if \mathcal{S}_h is contained in \mathcal{H}_r , $\mathbf{P}_2 \mathbf{x}$ can have a nonzero norm, depending on how \mathbf{x} is chosen (see Fig. 7 (b)). To remove the arbitrariness, we specify the distance to be zero when $\mathcal{S}_h \subset \mathcal{H}_r$, i.e. take the \mathbf{x} of case 1 in Fig. 7 (b). Based on these observations, we define the squared distance between the two subspaces as

$$(A.9) \quad d(\mathcal{H}_r, \mathcal{S}_h)^2 \equiv \min_{\mathbf{x} \in \mathcal{H}_r, \|\mathbf{x}\|=1} \|(\mathbf{P}_1 - \mathbf{P}_2)\mathbf{x}\|^2.$$

A.2 Interchangeability of subspaces. One interesting question here is whether the dual definition

$$(A.10) \quad d(\mathcal{S}_h, \mathcal{H}_r)^2 = \min_{\mathbf{x} \in \mathcal{S}_h, \|\mathbf{x}\|=1} \|(\mathbf{P}_2 - \mathbf{P}_1)\mathbf{x}\|^2$$

gives the same value as $d(\mathcal{H}_r, \mathcal{S}_h)$. Notice that $\mathbf{x} \in \mathcal{H}_r$ in Eq. (A.9) while $\mathbf{x} \in \mathcal{S}_h$ in Eq. (A.10).

To study the duality of the distance, first we express $(\mathbf{P}_1 - \mathbf{P}_2)^2$ with the bases of \mathcal{H}_r . With Eq. (A.8), the (i, j) element is calculated directly as

$$\mathbf{u}^{(i)\top} (\mathbf{P}_1 - \mathbf{P}_2)^2 \mathbf{u}^{(j)} = \delta_{i,j} - \sum_{l=1}^r K_{i,l} K_{l,j},$$

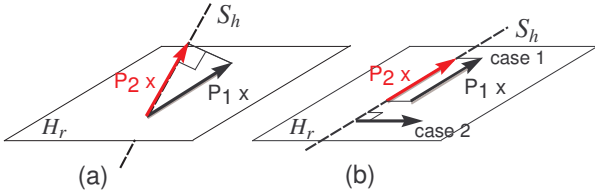


Figure 7: Comparison between \mathcal{S}_h and \mathcal{H}_r using the projection operators when $h = 1$.

where δ is Kronecker's delta, and $\mathbf{K} \in \mathbb{R}^{r \times h}$ is the Gram matrix whose (i, j) element is defined by

$$K_{i,j} \equiv \mathbf{u}^{(i)\top} \boldsymbol{\mu}^{(j)}.$$

In the matrix notation, we have $(\mathbf{P}_1 - \mathbf{P}_2)^2 = \mathbf{I}_r - \mathbf{K}\mathbf{K}^\top$, where \mathbf{I}_r is the r -dimensional identity matrix. Thus Eq. (A.9) becomes

$$(A.11) \quad d(\mathcal{H}_r, \mathcal{S}_h)^2 = 1 - \max_{\boldsymbol{\xi} \in \mathbb{R}^r, \|\boldsymbol{\xi}\|=1} \boldsymbol{\xi}^\top \mathbf{K}\mathbf{K}^\top \boldsymbol{\xi}.$$

Similarly, if we express $(\mathbf{P}_2 - \mathbf{P}_1)^2$ with the base of \mathcal{S}_h , the matrix representation will be

$$(\mathbf{P}_2 - \mathbf{P}_1)^2 = \mathbf{I}_h - \mathbf{K}^\top \mathbf{K},$$

so that Eq. (A.10) becomes

$$(A.12) \quad d(\mathcal{S}_h, \mathcal{H}_r)^2 = 1 - \max_{\boldsymbol{\eta} \in \mathbb{R}^h, \|\boldsymbol{\eta}\|=1} \boldsymbol{\eta}^\top \mathbf{K}^\top \mathbf{K} \boldsymbol{\eta}.$$

It is interesting to compare these two expressions. In Eq. (A.11), the maximizer is the top eigenvector of $\mathbf{K}\mathbf{K}^\top$, and is the same as the top left singular vector of \mathbf{K} . In Eq. (A.12), the maximizer is the largest eigenvector of $\mathbf{K}^\top \mathbf{K}$, or the top right singular vector of \mathbf{K} . Therefore, whether the left or right singular vector is the maximizer, the maximum value is the largest singular value of \mathbf{K} . Thus we have proved the following theorem.

THEOREM 2. For $\mathcal{H}_r \subset \mathbb{R}^w$ spanned by $\{\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(r)}\}$, and $\mathcal{S}_h \subset \mathbb{R}^w$ spanned by $\{\boldsymbol{\mu}^{(1)}, \dots, \boldsymbol{\mu}^{(h)}\}$,

$$(A.13) \quad d(\mathcal{H}_r, \mathcal{S}_h)^2 = d(\mathcal{S}_h, \mathcal{H}_r)^2 = 1 - \sigma_{\max}^2,$$

where σ_{\max} is the largest singular value of \mathbf{K} .

If $h = 1$, σ_{\max} is trivially computed as

$$(A.14) \quad \sigma_{\max} = \sum_{i=1}^r K(i, \boldsymbol{\mu})^2.$$

From Eqs. (A.13) and (A.14), we see that the CP score in Eq. (2.3) was defined as $z \equiv d(\mathcal{H}_r, \mathcal{S}_h)^2$. It is easy to verify $0 \leq z \leq 1$.

In closing Appendix, we explain why we chose the dimension of the feature space to be one (see Section 2). For example, consider the case $h = 2$. Suppose that \mathcal{S}_2 is spanned by $\boldsymbol{\mu}^{(1)}$ and $\boldsymbol{\mu}^{(2)}$, and that $\boldsymbol{\mu}^{(1)} = \mathbf{u}^{(1)}$ and $\boldsymbol{\mu}^{(2)} \perp \mathcal{H}_r$. In this case, the first column of \mathbf{K} has a single one and $r - 1$ zeros, and the second column is the zero vector. Therefore, σ_{\max} is one, giving the distance zero. However, the zero distance in this case is quite misleading since it does not at all mean either $\mathcal{S}_h = \mathcal{H}_r$ or $\mathcal{S}_h \subset \mathcal{H}_r$. Indeed, the distance is zero even if one of the bases is completely missing. If $h = 1$, this type of ambiguity does not appear. This is why we set the dimension of the test space to be one.

References

- [1] Esprit Project 26900, Technology for Enabling Awareness (TEA). 1998. [<http://www.omega.it/tea/>].
- [2] C. Chennubhotla and A. D. Jepson. Hierarchical eigensolver for transition matrices in spectral methods. In *Advances in Neural Information Processing Systems*, volume 17, pages 273–280, 2005.
- [3] N. de Freitas, Y. Wang, M. Mahdaviani, and D. Lang. Fast krylov methods for N-body learning. In *Advances in Neural Information Processing Systems*, volume 18, pages 251–258, 2006.
- [4] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.
- [5] G. H. Golub and C. F. van Loan. *Matrix computations* (3rd ed.). Johns Hopkins University Press, Baltimore, MD, 1996.
- [6] T. Idé and K. Inoue. Knowledge discovery from heterogeneous dynamic systems using change-point correlations. In *Proc. SIAM Intl. Conf. Data Mining*, pages 571–575, 2005.
- [7] E. Keogh and T. Foliás. The UCR time series data mining archive [<http://www.cs.ucr.edu/~eamonn/TSDMA/index.html>]. 2002.
- [8] V. Moskvina and A. Zhigljavsky. An algorithm based on singular spectrum analysis for change-point detection. *Communications in Statistics—Simulation and Computation*, 32(4):319–352, 2003.
- [9] H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes*. Cambridge University Press, 1989.
- [10] S. Roweis. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*, volume 10, pages 626–632, 1998.
- [11] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, volume 13, pages 682–688, 2001.
- [12] H. Zha and H. D. Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791, 1999.