Metodología de PCA

Una revisión de las estrategias de Análisis de Componentes Principales

PCA Methodologies

An overview on Principal Component Analysis

Esteban Vásquez Giraldo
FTA., Universidad EIA,
Envigado, Colombia
esteban.vasquez98@eia.edu.co

José Miguel Muñoz Arias
FTA., Universidad EIA,
Envigado, Colombia
jose.munoz25@eia.edu.co

***Resumen* — El análisis con PCA es una de las estrategias fundamentales más utilizadas en el ámbito del análisis estadístico de variables numéricas por su amplia cantidad de aplicaciones y versatilidad computacional. En el presente trabajo se desarrolla la intuición detrás de las metodologías de PCA, se presenta la estrategia fundamental para implementarlo con programación orientada a objetos (OOP) y aplicarlo a un caso de estudio. Finalmente se presentan otras variaciones útiles y aplicaciones recientes en complemento con una guía de uso con la librería SciKit-Learn.**

Palabras Clave - PCA, descomposición en valores singulares

***Abstract* — PCA analysis is one of the most widely used fundamental strategies in the field of statistical analysis of numerical variables due to its wide range of applications and computational versatility. This paper develops the intuition behind PCA methodologies, presents the fundamental strategy to implement it with object-oriented programming (OOP) and apply it to a case study. Finally, other useful variations and recent applications are presented in addition to a usage guide with the SciKit-Learn library.**

Keywords - PCA, singular Value Decomposition

I. Introducción

El Análisis de Componentes Principales (PCA) ha sido considerado como uno de los resultados más valiosos del álgebra lineal aplicada. El PCA es usado frecuentemente en todo tipo de análisis, desde neurociencia hasta gráficas de computadora, debido a que es un método no paramétrico relativamente simple el cual permite extraer información relevante de datos confusos.

Con una cantidad pequeña de trabajo adicional, el PCA provee una herramienta para reducir un conjunto de datos complejo a un espacio de menor dimensionalidad para revelar la dinámica simplificada del sistema, muchas veces escondida en la complejidad aparente de los datos.

II. Marco Teórico

A. *Intuición matemática*

En pocas palabras, se podría pensar en el PCA como una forma de encontrar la "base" más significativa para expresar un conjunto de datos aparentemente poco claro y confuso. La idea es obtener un nuevo sistema coordenado, cuyos ejes son las Componentes Principales (PCs), las cuales no tienen correlación entre ellas (es decir, forman una base de vectores ortogonales) pero en la cual cada una tiene la máxima correlación posible con los datos.

Cada se puede interpretar como un vector de un espacio m-dimensional, donde una dimensión representa una variable de las que dependen los datos. Por ejemplo, si se tiene un conjunto de datos con los ingresos y los egresos de n personas, se consiguen n datos (vectores) 2-dimensionales.

Por consiguiente, se puede llamar a esta base, la "base ingenua", ya que es la base en la cual se obtienen nuestros datos naturalmente pero puede no ser la más óptima. Pero ¿Cómo saber cuál es la base más óptima? Un criterio puede ser eliminar la redundancia entre las variables tanto como sea posible. Esto es exactamente a lo que se refieren las componentes principales. Una forma de cuantificar la redundancia entre las variables es calcular la covarianza, mientras menor sea la covarianza entre 2 variables, más "diferentes" son, es decir, más información le aportan, ya que si 2 variables tienen una alta correlación, será posible prescindir de una sin perder información de los datos.

La covarianza se define como:

$$\sigma_{AB}^2 = \sum_{i=1}^{n}(a_i - \mu_A)(b_i - \mu_B)$$

donde Ay Bson 2 variables de las que dependen los datos, $a_i$es el valor de la variable Apara el dato i, b_i es el valor de la variable Bpara el dato i y μ_Ay $\mu_B$son las medias de las variables A y Brespectivamente. Viendo esta definición, es claro que es conveniente realizar una estandarización de los datos ya que de esta manera, las medias se hacen cero y se obtiene una expresión vectorizada para el cálculo de la covarianza:

$$\sigma_{AB}^2 = \frac{1}{n}\mathbf{a}\mathbf{b}^T$$

De este modo se analizan todas las variables por pares, y se sintetizan todos los resultados en la llamada Matriz de Covarianza. Para realizar este cálculo de manera eficiente se define X como la matriz cuyas columnas serán las diferentes variables y las filas cada uno de los datos, es decir, será una matriz de dimensiones n x m. Para ello, la matriz de covarianza se toma como:

$$\Sigma = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

Es fácil ver cómo la matriz X^TX tiene en su entrada ij la covarianza de las variables i y j, y por tanto, contiene todas las correlaciones posibles entre todas las variables. Esta matriz resulta ser bastante especial, es una matriz cuadrada, real, simétrica y definida positiva. Entonces, si el objetivo es reducir la redundancia entre las variables, lo ideal es que cada variable tenga la menor correlación posible con todas las demás, esto quiere decir que lo ideal sería que los términos por fuera de la diagonal fueran cero. Así, el problema de remover la redundancia en las variables, se reduce a diagonalizar la matriz de covarianza. Por suerte, gracias a la forma especial de esta, siempre es posible encontrar una base de vectores ortogonales en la cual la matriz es diagonal. Los vectores de esta base tan particular son las anheladas Componentes Principales, y estos a su vez son los vectores propios de la matriz de covarianza. También se sabe que cada vector propio tiene asociado un valor propio, el cual en este caso será la varianza de los datos en la dirección del vector propio correspondiente.

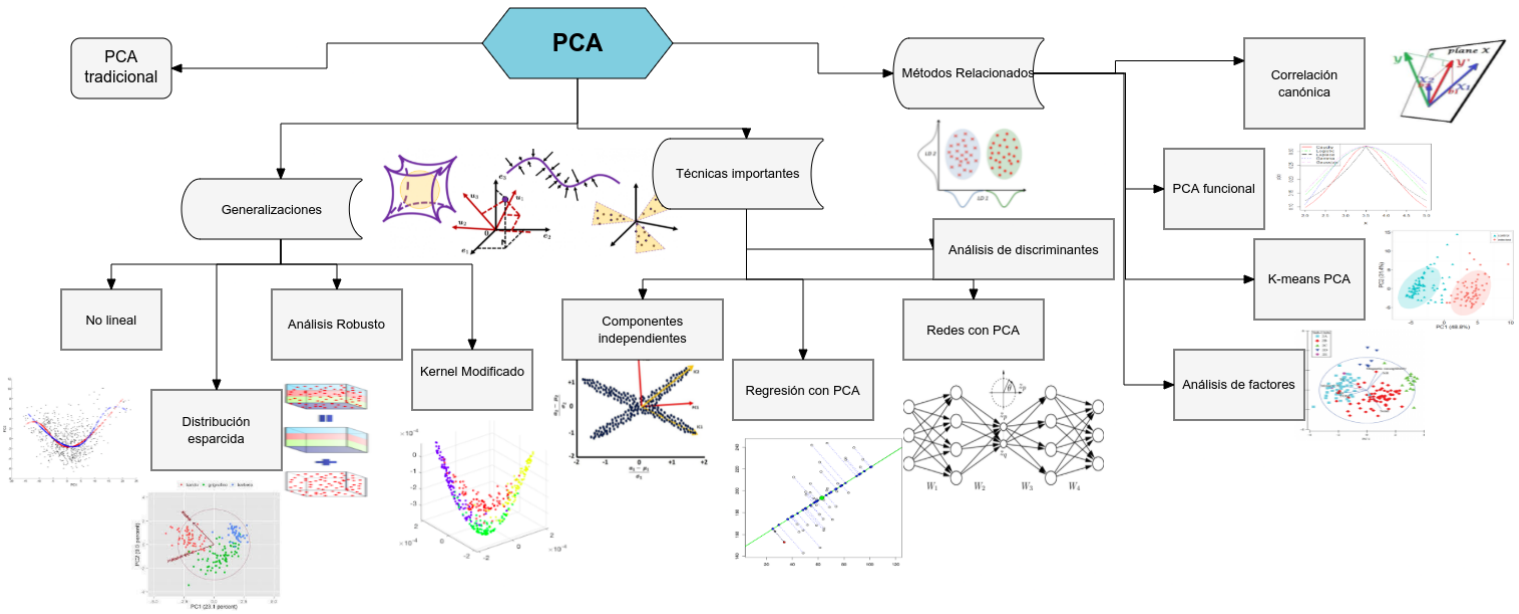
Lo que se logra con el PCA, es un conjunto de "nuevas variables" sin ninguna correlación entre ellas, es decir, un conjunto de variables que contienen la mayor cantidad de información de los datos. Esto es muy conveniente, porque como se mencionó antes, se puede prescindir de aquellas variables cuyo comportamiento está de alguna manera contenido en las demás. Al quitar algunas variables, se reduce la dimensionalidad de los datos, lo cual es muy útil para propósitos de compresión o visualización.

B. Métodos y modificaciones

Ya se construyó la intuición acerca de la matemática y geométrica que soporta el análisis con PCs. A continuación se presentan algunas estrategias y frameworks que potencian el PCA [1] mediante algunas variaciones, un mapa ilustrativo se ilustra en la Figura 1[1].

En cuanto a los métodos de generalización del PCA, cabe destacar que todo el análisis realizado con la matriz Σ se fundamenta en el concepto de distancia en un espacio euclidiano (con la idea del producto punto tradicional). No obstante, no siempre los datos representan distribuciones cartesianas, mucho menos cuando provienen de otros métodos como proyecciones temporales [2]. Para subsanar esta dificultad, se introdujeron los métodos de adaptación, algunos de los más versátiles son el PCA no lineal (que aprende en variedades no euclidianas). El PCA de distribución esparcida (que trata cada CP como una combinación lineal de las variables principales). El análisis robusto es fundamentalmente la descomposición de una matriz de datos en dos, una de las cuales contiene la información de los PC de mayor varianza (de reciente aplicación y prometedora versatilidad). El kernel modificado es directamente la aplicación de otra métrica para la minimización del error al ajustar cada componente mediante lo que se conoce como *kernel-trick*, los kernels auxiliares más importantes son los polinómicos, sigmoides y kernel de base radial (RBF) [3].

La búsqueda de optimizar computacionalmente los métodos y de adaptar el PCA para que dé cabida a acoples con otros algoritmos [4], ha llevado a la especialización de técnicas destacables que usan CPs. Se destaca por la optimización computacional el PCA incremental. que una baches (o si se quiere, mini-batch) para el entrenamiento de los modelos cuando los datos sean significativamente grandes, o el número de variables sea computacionalmente considerable. El acople de PCA con regresión es comúnmente usado y se denomina RPCA por siglas. Similarmente, la automatización de PCA antes del input a una red neuronal ha venido tomando fuerza, se le conoce como neural PCA (NPCA). Finalmente, la utilización de métodos bayesianos ha surgido en la literatura mediante un análisis de los discriminantes que surgen del método de PCA [5].



[1] Figura 1, mapa de metodologías con PCA

III. Metodología

A. Caso de Estudio

Para la utilización del PCA, se abordó el Dataset "DatosRiesgoCredito", fundamentalmente contiene información relacionada con aplicaciones a un Crédito bancario utilizando información personal, para la validación se extrajeron algunas variables específicas de mayor correlación con la reacción esperada de un modelo predictivo. Para una revisión de los datos, se llevó a cabo un Profiling, la revisión se presenta en la tabla I.

Número de variables Numéricas	6
Número de variables Categóricas	3
Número de observaciones	198
Datos vacíos o NaN	115 (6.5%)

Tabla 1. Profiling del Dataset.

Las variables numéricas extraídas son la edad (con media 40.7, MAD 9), el número de hijos (con moda 1 y MAD 1), el estrato (media 3 y MAD 1), ingresos y egresos del solicitante de crédito (ambas bien ajustadas a distribuciones □²) y el puntaje, la reacción objetivo de modelo predictivo buscado originalmente (bien ajustada a una distribución normal con media 48 y σ 287). La Figura 2 presenta la correlación de Pearson.

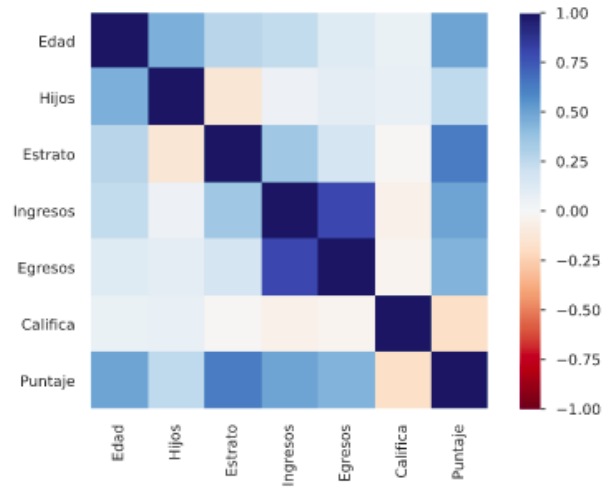


Figura 2. Mapa de calor para la correlación numérica.

Con las variables categóricas (Nivel de Estudio, Califica y Estado Civil) se llevó a cabo un Hot-Key encoding con el objetivo de poder analizar sus valores en el trabajo subsiguiente. Antes de ello, para no perder la relación discreta con el indicativo de si se realizó o no el préstamo, se analizó el potencial de Cramér, que se adjunta en la Figura 3.

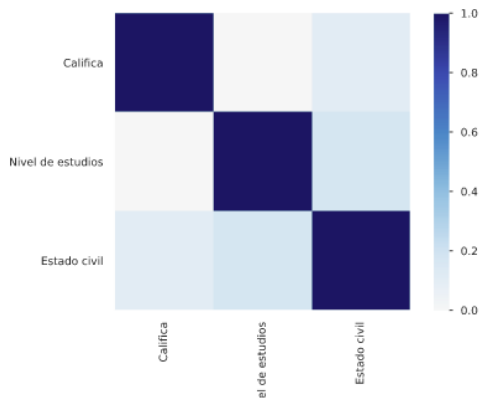


Figura 3. Mapa de calor para el potencial de Cramér.

B. Implementación en OOP

Con fines ilustrativos, se desarrolló la implementación del algoritmo de PCA en OOP en Python. La fundamentación para el algoritmo presentado en la Tabla 2 se puede obtener en la sección anterior.

1. Entradas: *Datos* como matriz de *m* observaciones x *n* variables, nombre de las columnas y el índice de cada observación.
2. Se normaliza cada columna de *Datos*.
3. Se calcula la matriz de covarianza ∑ y se extrae de la diagonalización los autovectores con sus respectivos autovectores.
4. Se ordena el vector con los autovalores y se mantiene otro vector con los argmax del vector original.
5. Se realiza la transformación lineal al espacio de los autovectores mediante *Datos^T · autovectores.*
6. Se grafican con scatter los datos transformados y se representan los autovectores estandarizados con el vector de autovalores.

Tabla 2. Algoritmo para PCA

Concretamente, el algoritmo desarrolla 4 métodos analíticos para la clase PCA: normaliza() que utiliza métodos directos de Pandas. El método `eigen_vectores`() que toma la covarianza y calcula los autoespacios y `transformacion_lineal`() que lleva los datos al espacio de los dos componentes principales de mayor peso. Todos ellos se llaman en el método `ajustar`(). Adicionalmente, se incluyen los métodos `graficarResultado`() que hace un scatter de los datos sobre el espacio transformado, `graficaInteractiva`() que permite una interacción con la gráfica mediante Plotly y finalmente el método `graficarIndiceExplicacion`() que imprime el porcentaje de la varianza que se explica como función del número de componentes en que realice una descomposición.

C. Otras Librerías

Por completez, es necesaria la revisión de librerías que realizan de forma sistemática el PCA, que tengan más versatilidad para la implementación de las modificaciones mencionadas en la sección anterior y cuyos *output* sean input default para otras clases de las mismas librerías. En este aspecto, hay diversas librerías, a continuación, se mencionan tres módulos estándares de ML y se hace especial énfasis en los métodos de SciKit-Learn.

- ☐ **TensorFlow [6]:** Recordando que esta librería utiliza fundamentalmente APIs de alto nivel como Keras, esta plataforma tiene implementado el módulo `ttf` con dos clases: `TFTransformOutput` y `TransformFeaturenLayer`. Para llamar el PCA, basta hacer `ttf.pca(un_tensor, dimensiones_deseadas_output, dType)`. El output de esta función es un tensor con el número de componentes indicado.
- ☐ **PyTorch [7]:** Igualmente, este framework también implementa el PCA como un método de la clase

`torch` (tensores de bajo rango), para llamarlo se utiliza `torch.pca_lowrank(un_tensor, niter=2)`. El método usado para la reducción de dimensión es estocástico, esto marca una diferencia en el tiempo computacional con las otras librerías.

- **SciKit-Learn [8]:** Es el API usado por excelencia debido a facilidad de uso y versatilidad. Dentro del módulo `sklearn.decomposition` (con diferentes métodos para descomposición de matrices) se tiene la clase `PCA(numero_componentes,svd_solver)`. Para el uso directo, se genera un objeto con el llamado `pca = PCA(n_components)`. Posteriormente y se ajusta a la matriz de observaciones mediante `pca.fit(matriz_numpy)`.Para acceder a los vectores descompuestos: `pca.components_`. Los datos correspondientes a la descomposición en valores singulares y la varianza explicada se pueden obtener con los atributos `singular_values_` y `explained_variance_ratio_` respectivamente.

  Lo más destacable del uso de la librería sklearn es que tiene bien implementadas varias de las modificaciones al PCA, por ejemplo algunas de ellas, con sus respectivos métodos son:
  - PCA esparcido (`decomposition.SparsePCA`)
  - Incremental (`decomposition.IncrementalPCA`)
  - Kernel modificado (`decomposition.KernelPCA`)
  - PCA factores (`decomposition.factorAnalysis`)
  - Descomposición en valores singulares (`decomposition.TruncatedSVD`).

No obstante, estas no son las únicas librerías que implementan PCA, al punto que la gran mayoría de los productos científicos relacionados realizan sus propias implementaciones. Un claro ejemplo fue el desarrollo del Robust PCA, implementado por [9] en el 2009.

Por completez, se realizó la implementación de la librería sklearn para el caso de estudio abordado en este trabajo, haciendo uso del ajuste, transformación y transformación inversa. Los resultados, en contraste con los obtenidos en la clase PCA implementada en OOP se presentan a continuación.

IV. Análisis de Resultados

El estudio de los resultados obtenidos por el PCA requieren el tratamiento específico de la reducción de dimensionalidad. Por este motivo, se han venido desarrollando en la literatura diversos métodos para evaluar los ajustes y proyecciones, principalmente para seleccionar el número de componentes ideal para realizar la descomposición [10]. Una de las principales métricas consiste en el error de reconstrucción de la varianza (VRE), que evalúa la similitud de las componentes proyectadas contra las originales como coeficiente de r2 y el error cuadrático medio. Por consiguiente, al desarrollar el PCA, se busca el número de componentes que expliquen la varianza hasta el nivel de tolerancia deseado.

En el código propio, este método fue implementado en la función `graficarIndiceExplicacion()`. Simultáneamente se implementó en el notebook con sklearn. Ambos resultados se ilustran en la figura 4, donde es evidente que ambos modelos están bien comportados y muestran la misma tasa de crecimiento por variables incluidas.

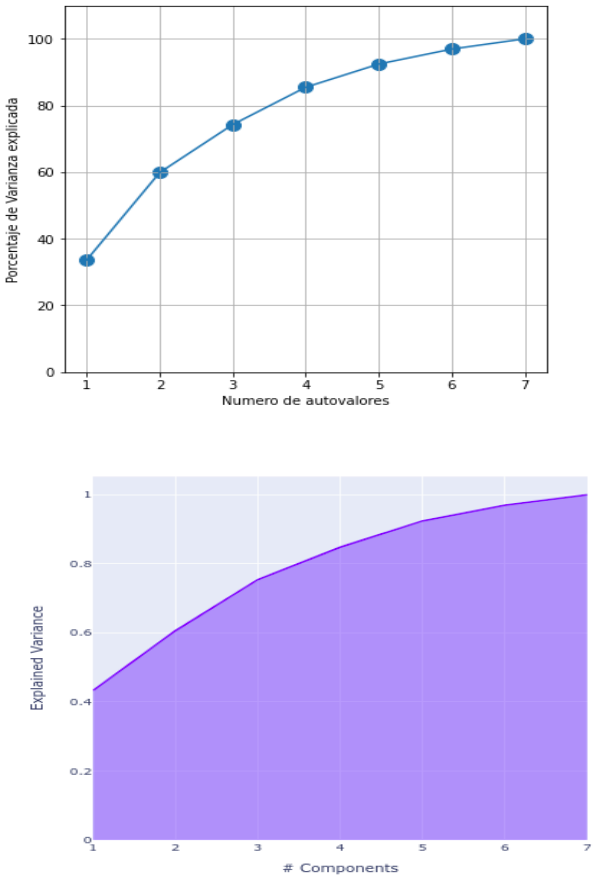


Figura 4. VRE para el PCA propio (arriba) y sklearn (abajo).

Similarmente, una forma de ratificar que ambos modelos funcionen es la graficación en los espacios transformados. Esta estrategia permite obtener una visualización de las posiciones respectivas de cada punto en cada una de las variables, (cuyo objetivo es directo hacer comprensible el PCA en solo dos componentes). La figura 5 ilustra esta obtención para el caso de la clase propia, donde el color codifica la variable objetivo.

Se realizó un estudio del tiempo de cómputo asociado a cada una de las implementaciones en el ajuste para reducción de dimensionalidad. La implementación propia tomó 9.21 ms en ajustar los datos del caso de estudio y 36.5 ms para el Iris DataSet. En contraparte, la implementación en Scikit-Learn tomó 939 µs en ajustar el PCA en todas las componentes y 21.9 ms con el Iris DataSet. La diferencia en tiempo de cómputo, a pesar de ser considerable, está siendo amortiguada por tener la implementación como una clase.

Ambos programas pueden ser consultados en [12].

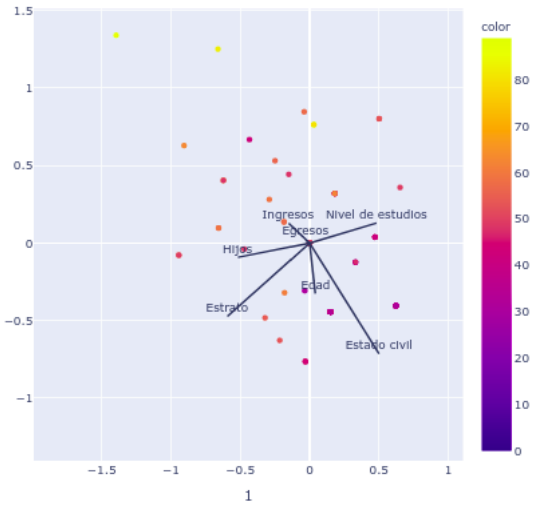


Figura 5. Proyección en 2 componentes para PCA.

V. Prospectos Futuros

Dentro del desarrollo de este proyecto se encontró que es interesante la comparación de los métodos y métricas para la evaluación de diferentes implementaciones de PCA. Para el desarrollo futuro queda la revisión en regresión comparativa antes y después de las proyecciones y tiempos computacionales. Adicionalmente, la puesta en funcionamiento de estrategias alternativas como kernel y PCA multicomponente en la clase propia permitirá una comprensión más profunda de estos métodos. Para finalizar, sería también interesante buscar estrategias para llevar a cabo el PCA robusto, que ha demostrado ser una de las estrategias más versátiles para la limpieza de datos numéricos, imágenes y hasta aplicaciones físicas como descomposición de movimiento de fluidos en armónicos temporales [6] en Sk-learn (aún no implementado). Este campo se muestra ampliamente prometedor, no solo en la limpieza y manejo de datos, sino también en las aplicaciones a fenómenos físicos dignos de estudio.

VI. Conclusiones

Fue posible el estudio del método de PCA mediante un caso de estudio concreto de análisis de riesgo de crédito. Este aproximamiento permite obtener una idea intuitiva sobre la metodología y un framework para realizar la reducción de dimensionalidad, adicionalmente se exponen algunos de los métodos más sofisticados que se basan en PCA.

La importancia de este recorrido muestra la posibilidad de hacer transparente este método y quitar el estatus de caja negra. Adicionalmente fue posible la realización del método como clase en OOP. Comparativamente se utilizó SciKit-learn para el mismo caso de estudio, el análisis mediante VRE de los resultados para ambos modelos son compatibles, no obstante, la eficiencia computacional se ve superada. Finalmente, se realizó un recorrido por la implementación de los métodos discutidos en la librería SciKit-learn.

Referencias Bibliográficas

1. Jolliffe, I. T. (2002). Generalizations and adaptations of principal component analysis. *Principal Component Analysis*, 373-405.
2. Bandeira, A. S., Kunisky, D., & Wein, A. S. (2019). Computational hardness of certifying bounds on constrained PCA problems. *arXiv preprint arXiv:1902.07324*.
3. Scholkopf, B. (2001). The kernel trick for distances. *Advances in neural information processing systems*, 301-307.
4. Arora, R., Cotter, A., Livescu, K., & Srebro, N. (2012, October). Stochastic optimization for PCA and PLS. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (pp. 861-868). IEEE.
5. Zheng, W., Lin, Z., & Wang, H. (2013). L1-norm kernel discriminant analysis via Bayes error bound optimization for robust feature extraction. *IEEE transactions on neural networks and learning systems*, *25*(4), 793-805.
6. Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., ... & Saurous, R. A. (2017). Tensorflow distributions. *arXiv preprint arXiv:1711.10604*. https://www.tensorflow.org/tfx/transform/api_docs/python/tft
7. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.https://pytorch.org/docs/stable/generated/torch.pca_lowrank.html
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA
9. Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. *Journal of the ACM (JACM)*, *58*(3), 1-37.
10. Mnassri, B., Ananou, B., & Ouladsine, M. (2010, June). A generalized variance of reconstruction error criterion for determining the optimum number of principal components. In *18th Mediterranean Conference on Control and Automation, MED'10* (pp. 868-873). IEEE.
11. Scherl, I., Strom, B., Shang, J. K., Williams, O., Polagye, B. L., & Brunton, S. L. (2020). Robust principal component analysis for modal decomposition of corrupt fluid flows. *Physical Review Fluids*, *5*(5), 054401.
12. Repositorio del proyecto. E Vasquez, J Muñoz. https://github.com/munozariasjm/PCA_ML_EIA