

Visualizador radial.

Interfaz gráfica.

José Manuel Vera Lillo | Máster en Data Science & Big Data, U-tad | 15/09/2019

**Índice.**

* Objetivos.
* Introducción teórica.
* Uso de la interfaz.
* Librerías implementadas.
* Construcción de la interfaz
* Conclusiones.
* Bibliografía.

**Objetivos.**

En este trabajo se pretende realizar una interfaz gráfica usando Python, la cual se capaz de realizar visualizaciones de diferentes métodos de reducción de la dimensionalidad, así como su interpretación radial.

También se pretende conseguir que mediante la interacción del usuario con la gráfica (por pantalla), esta sea capaz de representar radialmente los datos al gusto del usuario.

**Introducción teórica.**

Para la visualización de datos en la interfaz gráfica, se siguen diferentes procesos. Primero hemos de seleccionar la forma en la que se visualizarán los vectores, los cuales de una forma gráfica) nos indican la importancia de las variables seleccionadas con los datos representados.

Estos vectores se pueden representar de tres formas diferentes:

* **PCA**: Análisis del componente principal, este método sirve para redimensionar la matriz de datos, de tal forma que nos dará como resultante una matriz con N filas (tantas como variables hayamos seleccionado) y dos columnas (r y θ), este se caracteriza por realizar una combinación lineal, el cual proyecta los datos, sin perder todas sus propiedades para describirlos en un espacio de baja de dimensión1.
* **LDA:** Análisis discriminante Lineal, lo utilizaremos de igual forma que PCA, para obtener una matriz con N filas y dos columnas, pero en este método se obtienen resultados notoriamente efectivos en separabilidad intraclases, su fortaleza radica en proyectar un buen desempeño en tareas de reducción de dimensión conservando la separabilidad entre clases. LDA no posee un buen rendimiento en tareas de compresión como PCA, porque desarrolla un proceso diferente el cual busca seguir patrones de distribución gaussianas para un buen funcionamiento, que garantice una transformación intraclase para hallar la máxima dispersión entre cada clase y a su vez, reducir a su mínimo posible la dispersión interna de cada clase. LDA tiene un mejor rendimiento en tareas de clasificación gracias al punto de corte del modelo discriminante que proyecta una robusta regla de decisión. PCA posee un mejor desempeño que LDA en tareas de compresión de datos, sin embargo, no es eficiente realizando tareas de separación entre clases1.
* **Interactiva:** También después de haber seleccionado uno de los métodos posteriores, se podrán mover estos vectores de forma interactiva de forma que seremos nosotros los que dotaremos a las variables de la importancia que queramos desplazándolas por la gráfica.

Posteriormente a la selección de los vectores, el programa usará estos para dibujar los datos en consideración a estos. Para ello debemos seleccionar el algoritmo con el que dibujarlos.

* **SC:** Este algoritmo es simplemente una multiplicación matricial entre la matriz de datos X y la matriz de vectores V.
* ***RadViz*:** Muestra todos los atributos como puntos anclados al perímetro de una circunferencia, y separados en forma equidistante dependiendo de la cantidad de atributos. Toma como base al paradigma del tensor proveniente de la física de partículas, puntos de la misma clase se atraen entre sí, los de diferente clase se repelen entre sí, y las fuerzas resultantes se ejercen sobre los puntos de anclaje. Una ventaja de RadViz es que conserva ciertas simetrías de los datos, y su principal desventaja es la superposición de puntos8.
* ***Adaptable9*:** Este algoritmo, puede considerarse como una extensión de SC, con el añadido de que las proyecciones se definen a través de problemas de aproximación de normas convexas. En particular, el enfoque también minimiza los errores que los usuarios cometerían al aproximar valores de atributos de alta dimensión proyectando puntos mapeados ortogonalmente sobre los ejes. Sin embargo, incorpora elementos y variantes que no sólo ofrecen a los usuarios un conjunto más rico de proyecciones para explorar, pero también facilita varias tareas de análisis de datos.
* ***Adaptable* *exact9*:** Es una extensión del algoritmo *Adaptable*, donde la restricción estricta que obliga a realizar estimaciones exactas para un atributo puede utilizarse para ordenar los datos correctamente según el eje correspondiente. Sin embargo, puede aumentar considerablemente los errores de estimación en las otras variables.
* ***Adaptable* *ordered9*:** Es una extensión del algoritmo *Adaptable*, donde se aplicada el problema de optimización para representar los datos perfectamente, y por lo tanto correctamente ordenados, según el eje. A diferencia de *Adaptable exact* que, puede aumentar los errores de estimación en las otras variables considerablemente. Y con el fin de aliviar este problema, en esta sección se considera una restricción más leve que sólo requiere que los puntos sean ordenados correctamente a lo largo del eje.

A continuación, se expondrá de una forma visual, como afectan los diferentes algoritmos a la hora de graficar con los métodos PCA y LDA.

Imagen que contiene texto, mapa

Descripción generada automáticamenteAsí se verían los diferentes algoritmos para la gráfica de dispersión con los vectores propios de PCA en el *dataset* de iris:

Así se verían los diferentes algoritmos para la gráfica de dispersión con los vectores propios de LDA en el *dataset* de iris:

Imagen que contiene texto, mapa

Descripción generada automáticamente

**Uso de la interfaz.**

En primera instancia, se debe cargar un archivo, para ello se puede arrastrar y soltar el fichero encima del botón *drag and drop or select files* o bien se puede clicar en este y aparecerá una pantalla emergente de nuestra carpeta de documentos, donde podemos seleccionar el archivo a utilizar.

Una vez cargado el archivo se debe clicar el botón de *propagate data*, para propagar los datos a los selectores.

Cuando ya estén los datos cargados correctamente, se deben seleccionar los parámetros bajo los cuales se quiere realizar la visualización, es decir seleccionar tanto las variables (columnas) a representar, como el algoritmo de inicialización o el método radial por el cual se representarán los puntos de la gráfica de dispersión, por defecto en estos dos últimos parámetros viene seleccionado tanto PCA como algoritmo inicializador y SC como método de representación radial. Hay métodos (Adaptable, Adaptable exact y Adaptable ordered) que requieren un parámetro adicional que es la norma del vector, en caso de seleccionar alguno de estos métodos radiales, se desplegará un nuevo *RadioItem* que nos permitirá seleccionar este (1, 2 o Inf).

También, hay que seleccionar una variable para ordenar los datos según esta, donde haremos de nuevo uso de un desplegable para escoger dentro de las variables anteriormente seleccionadas.

Por último, seleccionamos la clase, *target*, mediante un desplegable e indicamos como queremos que represente los colores, por valor o por *target* haciendo uso de un *RadioItem.*

Una vez tengamos todos estos parámetros definidos, al clicar el botón *calculate*, la interfaz muestra por pantalla la visualización, donde se puede hacer uso de la interactividad de la gráfica.

Para ello debemos coger el extremo superior del vector representado y desplazarlo a la nueva posición que deseemos, y el programa se encargará de volver a realizar los cálculos con los nuevos vectores y representando de nuevo con esos valores la gráfica de dispersión. También se puede seleccionar el rango de valores por el cual queramos que la gráfica muestre los datos. Es decir, podemos variar el radio de los datos de entrada para quedarnos únicamente con los de mayor, menor o valor initermedio.

**Librerías implementadas.**

Tanto para la realización de la interfaz como para la implementación de la función *mapping* en *calculate\_mapping\_general* se usan las siguientes librarías.

* Numpy.
* numpy.matlib.
* scipy.optimize, linprog .
* cvxpy == 0.4.2.
* plotly == 4.0.0.
* plotly.graph\_objects.
* dash.dependencies.
* pandas.
* scipy, stats.
* sklearn.decomposition, PCA.
* sklearn.discriminant\_analysis, LinearDiscriminantAnalysis.
* sklearn.preprocessing, LabelEncoder.
* json.
* operator.
* os.
* base64.
* io.

**Construcción de la interfaz.**

La construcción de la interfaz se ha realizado con la librería *dash-plotly* (versión 4.0.0), la cual permite realizar una aplicación web, ya que aúna tanto Python para realizar los cálculos, como html5, css y javascript para la realización de la interfaz web.

Dentro de la carpeta del programa se encuentran diferentes archivos que describen la interfaz, en primer lugar se encuentra el archivo Interfaz Dash que es el elemento principal que contiene a las otras funciones, también se encuentra calculate\_mapping\_general que es el encargado de realizar los cálculos de los algoritmos para el mapeo radial, otro archivo que contiene es common\_func, el cual contiene la función que realiza las visualizaciones, por último, dentro de la carpeta *assets*, se encuentran dos archivos más, *header* que es un archivo *css* que contiene los estilos y márgenes de la interfaz y *custom-script,* un archivo *javascript*, que se encarga de redireccionar los estilos del *html* creados en *dash-plotly.*

En primer lugar, se describen los diferentes componentes que se visualizarán vía web:

* **dcc.Upload2:** permite introducir por pantalla los archivos que deseemos subir a la aplicación.
* **dcc.Dropdown3:** aquí se describen los diferente desplegables que se usan en la aplicación para la selección de diferentes componentes, ya puedan ser tanto la selección de variables, como el método de obtención de los vectores, o la selección del *target*.
* **dcc.RadioItem4:** de igual forma que los desplegables, se puede hacer uso de los radioItems, estos se han usado para la selección del algoritmo que representa los datos, como para la elección del color en la gráfica (por valores o según el *target*).
* **Dcc.Graph6:** es el elemento *html* de *dash-plotly*, que permite visualizar las gráficas.

Aquí podemos ver una muestra de cómo se declaran los elementos descritos arriba en *dash-plotly:*

Imagen que contiene captura de pantalla, texto

Descripción generada automáticamente

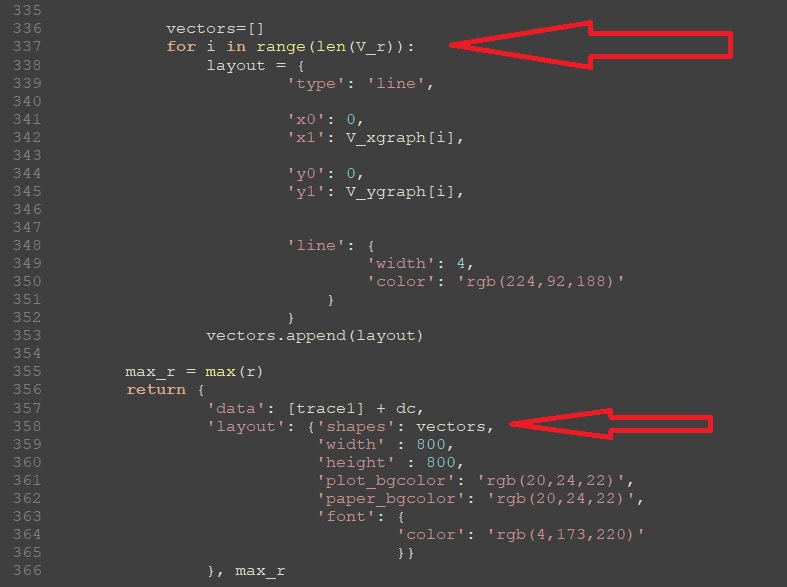
Una vez seleccionados los elementos que se visualizarán en la interfaz se procede a trabajar con los valores proporcionados por estos, para ello y dado que *dash-plotly* no permite la declaración de variables globales, este proceso se debe realizar mediante la utilización de *callbacks*, para ello hay que declarar tanto las variables de entrada a la función, como las salidas.

Aquí una muestra de cómo se declaran los *callbacks* en *dash-plotly*:

Imagen que contiene captura de pantalla, texto

Descripción generada automáticamente

El uso de los *callbacks*, se declaran haciendo uso de un *app.route*, donde en primera estancia se definen las entradas y salidas, en cada entrada y salida, hay que definir a su vez tanto la id, como el tipo de valor que se va a retornar, en esta interfaz podemos entrar diferentes valores, *data* para los datos, *options*, para las opciones de los *dropdown* y *RadioItem*, *value* para escoger el valor seleccionado de estos y *relayoutData* para el diccionario en formato *json* (posteriormente se define para que se usa este diccionario). Una vez declarados los *callbacks5* se le asocia una función, de las cuales podemos encontrar:

* En primer lugar, se declara la función *parse\_components*, esta función recoge el archivo introducido en pantalla (ya sea excel o csv) y lo devuelve a la función *update\_output*, el cual recoge el archivo y lo transforma en un diccionario.
* Una vez tenemos los datos en forma de diccionario, se le pasan a la función *update\_filter\_*columns, la cual proporciona las diferentes variables (columnas) que posee el archivo de datos para devolvérselos a la función *set\_graphics\_selector*s donde se almacenan las variables seleccionadas.
* De forma parecida trabajan las funciones *choose\_vectors\_options* y *choose\_vectors\_value*, donde primero dentro de las variables a seleccionar, nos proporciona un listado de las variables a seleccionadas para poder escoger una, estas funciones no son necesarias para todos los casos, solo se necesitarán cuando se trabaje con los algoritmos Adaptable, Adaptable exact y Adaptable ordered, ya que necesitan una variable dominante en el proceso de cálculo.
* También se describen las funciones *set\_vector\_norm* que proporciona la norma bajo la que trabajarán los algoritmos y *choose\_target* que nos permite seleccionar el *target* o clase de los datos, esto es necesario tanto para el proceso de cálculo de los vectores con el método LDA, ya que necesita la clase a la que pertenecen los datos para poder realizar la separación intraclases anteriormente citada, como para posterior visualización de datos, ya que estos se colorearán en función de dos opciones o por valor o por *target* (clases).
* La función *display\_selected\_data* recoge la posición a la que se han desplazado los vectores, para devolvérselos a la función *display\_graph* encargada de la realización de los gráficos.
* ***display\_graph6*:** en esta función es donde se recogen todas las selecciones realizadas por el usuario, esta en un primer momento (antes de clicar el botón de *calculate*) devuelve una gráfica vacía. Pero una vez realizadas todas las selecciones por parte del usuario, al clicar el botón de *calculate*, estas pasan a la función *commonFunc*, en esta función se declaran otras dos funciones, *get\_index* que nos proporciona los índices de las variables para poder trabajar con ellas y e *interact* que determina si los vectores han cambiado de posición de forma interactiva (si lo hemos desplazado nosotros manualmente se actualizará la gráfica, sinó no volverá a realizar los cálculos).
* ***commonFunc:*** esta función realiza tanto los cálculos como las graficas a visualizar, una vez se han seleccionado todos los procesos, esta hace uso de la función *calculate\_mapping\_general* (donde se calcula la matriz de datos a representar), otro inconveniente de *dash-plotly* es que solo permite realizar cambios de forma interactiva en gráficas cartesiana, por lo que se hace una reconversión de los datos de coordenadas polares a cartesianas, una vez tratados los datos de forma conveniente se le proporcionan a las gráficas, haciendo uso de *go.Scatter*, también se añaden los vectores de dos formas diferentes, una es en forma de datos, para poder visualizar los nombres de las variables en la gráfica y la otra es en forma de *shape*, esta segunda forma es necesaria, ya que es la que nos permitirá desplazarlos a través de la gráfica y recoger los puntos a donde hemos desplazados los vectores. Previamente a la visualización de los datos se deben calcularas los vectores propios de los métodos PCA y LDA. Para ello se hace uso de *sklearn.decomposition*, *PCA* y *sklearn.discriminant\_analysis*, *LinearDiscriminantAnalysis* que nos permite calcular las redimensiones de los datos con *pca.fit\_transform(X\_std)* y posteriormente obteniendo los vectores propios de la forma *V\_r = np.transpose(pca.components\_[0:2, :])* para PCA y de forma análoga para LDA con *lda.fit(X\_std,targ)* y *V\_r = lda.scalings\_*, cabe señalara que para el cálculo de LDA se le debe pasar también una columna con el *traget*. Una vez se han obtenido los vectores propios de PCA o LDA, se hace uso de estos en la función *mapping* de *calculate\_mapping\_general*, la cual proporciona una matriz P de dos columnas (x e y) de los datos a representar.
* **Actualización de la gráfica por desplazamiento del vector:** primero para poder hacer que los vectores se puedan desplazar por la gráfica, se deben declarar los vectores no como líneas dentro de la gráfica, sino como *shapes* dentro del *layout* (esto se encuentra dentro de common\_func).

Una vez declarados los vectores como *shapes*, debemos indicarle a *dcc.Graph* que estos puedan ser editables, para ello se añade el comando *'editable': True* y *'edits': {'shapePosition': True }*.

Imagen que contiene captura de pantalla

Descripción generada automáticamente-

Una vez se le ha indicado a *dcc.Graph* que los vectores son editables, estos ya se pueden desplazar por toda la gráfica, pero ese no es el comportamiento esperado, ya que solo interesa que se puedan mover radialmente, es por eso que hay que hacer uso de *javascript7* para declarar una regla que impida a estos desplazarse en todas direcciones, ya que *dash-plotly* incorpora por defecto una serie de estilos y clases, los cuales prevalecían a pesar de modificarlos con css7.

Imagen que contiene texto

Descripción generada automáticamente

Con esta regla de *javascript*, se pretende anular la acción por la cual se puede desplazar el vector entero y solo se pueda desplazar el extremo. Para ello *javascript* busca dentro del *html* el *style* que permite esta acción y la cancela, esto no es un regla permanente, por ello se define *setInterval(clean, 1000)* con lo que se le dice que a cada segundo busque dentro del *html* si este *style* está funcionando para cancelarlo. Con esto se consigue que los vectores solo se desplacen radialmente.

Imagen que contiene texto, captura de pantalla

Descripción generada automáticamentePara la posterior actualización de los datos, se declara una nueva función dentro de *dash-plotly* que recoge la posición a la que se ha desplazado el vector y devuelve los valores en un diccionario *json* a la función *display\_graph*.

Este diccionario lo recoge la función *display\_graph* que hace uso de *commonFunc* donde inicialmente se declara que los valores recogidos por el vector estén vacíos (axis\_data).

Imagen que contiene dispositivo

Descripción generada automáticamente

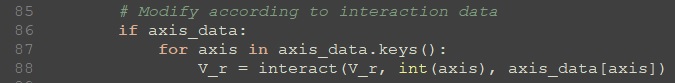
Para posteriormente, si se dispara su uso, este los implemente dentro de *commonFunc.*

Imagen que contiene texto

Descripción generada automáticamenteDonde se hace uso de la función *interact*, que recoge los valores almacenados:

**Conclusiones.**

Como conclusiones, podemos observar que los objetivos han sido cumplidos, ya que se ha conseguido realizar una interfaz que visualiza radialmente las diferentes variables seleccionadas por el usuario, así como aplicar los métodos de reducción de la dimensionalidad PCA y LDA. También se ha conseguido dotar a la misma de una gran interactividad, ya que es capaz de actualizar las gráficas interactuando por pantalla con ellas, haciendo uso del movimiento de los vectores.

Por lo tanto, puede servir como referente en el futuro, en el uso de gráficas en interfaces que necesitaran el uso de gráficas radiales con Python, así como su uso interactivo en otras aplicaciones de *dash-plotly*.

**Bibliografía.**

1. LDA versus PCA – Diego Peluffo. <http://diegopeluffo.com/publicaciones/2016_INCISCOS_LDA%20versus%20PCA.pdf>
2. <https://dash.plot.ly/dash-core-components/upload>
3. <https://dash.plot.ly/dash-core-components/dropdown>
4. <https://dash.plot.ly/dash-core-components/radioitems>
5. <https://dash.plot.ly/getting-started-part-2>
6. <https://dash.plot.ly/dash-core-components>
7. <https://dash.plot.ly/external-resources>
8. <https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-33052018000100130>
9. Adaptable Radial Axes Plots for Improved Multivariate Data Visualization, M. Rubio-Sánchez, A. Sanchez and D. J. Lehmann <http://www.lite.etsii.urjc.es/manuel.rubiosanchez/papers/Eurovis2017_ARA_plots.pdf>