Significado de `singular_values_`

El atributo `singular_values_` se refiere a los valores singulares de la matriz de datos *centrada* $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top - \boldsymbol{\mu}^\top \\ \vdots \\ \mathbf{x}_n^\top - \boldsymbol{\mu}^\top \end{bmatrix} \in \mathbb{R}^{n\times 784}, \qquad \text{donde} \qquad \boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \ , \tag{1}$$

Es decir, es la matriz que tiene, en cada fila, cada una de las imágenes $\mathbf{x}_i \in \mathbb{R}^{784=28\cdot 28}$ (puestas en forma de vector) del dataset *centradas*, es decir, restándoles a cada una la media $\boldsymbol{\mu} \in \mathbb{R}^{784}$.

Definiendo la *descomposición en valores singulares* (SVD) de $\mathbf{X}$, con las dimensiones de cada matriz como:

$$\underbrace{\mathbf{X}}_{n\times d} = \underbrace{\mathbf{U}}_{n\times n}\underbrace{\mathbf{D}}_{n\times d}\underbrace{\mathbf{V}^\top}_{d\times d} \ , \tag{2}$$

Significa que el atributo `singular_values_` se corresponde con los d valores de la diagonal principal de $\mathbf{D}$.

Esto lo podemos verificar con el siguiente código:

```
>_

import numpy as np
from sklearn.decomposition import PCA

# random data
X = np.random.rand(2_000, 1_000)   # (n, d)

# fit PCA
pca = PCA()
pca.fit(X)

# Manually compute singular values of the *centered* data matrix:
svals_data = np.linalg.svd(X - X.mean(axis=0, keepdims=True), compute_uv=False)

# singular_values_ == svals_data?
print(np.allclose(pca.singular_values_, svals_data)) # prints True
```

Relación de `singular_values_` con `explained_variance_`.

Por un lado, sabemos que `explained_variance_` se corresponde con la varianza de los datos en las direcciones principales. Dicho de otra forma, el atributo `explained_variance_` contiene los *valores singulares de la matriz de covarianza de* $\mathbf{X}$.

Esta matriz de covarianza, $\boldsymbol{\Sigma}$, la podemos calcular como:

$$\boldsymbol{\Sigma} = \frac{1}{n-1}\,\mathbf{X}^\top\mathbf{X} \ , \tag{3}$$

Sabiendo esto, ¿sabrías relacionar los valores singulares de $\boldsymbol{\Sigma}$ con los de $\mathbf{X}$? (te animo a que lo pruebes a sacar la relación 😉).

Debido a que $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ (Eq. 2), significa que la matriz de covarianza $\mathbf{\Sigma}$ también la podemos calcular como:

$$
\begin{aligned}
\mathbf{\Sigma} &= \frac{1}{n-1}\ \mathbf{X}^\top\mathbf{X}\ , && \text{Por Eq. 3}\ , && (4)\\
&= \frac{1}{n-1}\ (\mathbf{U}\mathbf{D}\mathbf{V}^\top)^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top\ , && \text{Por Eq. 2}\ , && (5)\\
&= \frac{1}{n-1}\ \mathbf{V}\mathbf{D}^\top\mathbf{U}^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top\ , && && (6)\\
&= \frac{1}{n-1}\ \mathbf{V}\mathbf{D}^\top\mathbf{D}\mathbf{V}^\top\ , && \text{Ya que } \mathbf{U}^\top\mathbf{U} = \mathbf{I} \text{ por ortonormalidad}\ . && (7)
\end{aligned}
$$

Acabamos de llegar a la descomposición en valores singulares (SVD) de $\mathbf{\Sigma}$:

$$
\underbrace{\mathbf{\Sigma}}_{d\times d} = \underbrace{\mathbf{V}}_{d\times d}\underbrace{\left(\frac{1}{n-1}\ \mathbf{D}^\top\mathbf{D}\right)}_{d\times d}\underbrace{\mathbf{V}^\top}_{d\times d}\ , \tag{8}
$$

Ya que la matriz $\frac{1}{n-1}\ \mathbf{D}^\top\mathbf{D}$ es diagonal, y la matriz $\mathbf{V}$ es ortonormal.

Por tanto, y como conclusión, existe la siguiente relación entre `singular_values_` y `explained_variance_`:

$$
\texttt{explained\_variance\_} = \frac{1}{n-1}\ (\texttt{singular\_values\_})^2\ , \tag{9}
$$

La cual, la podemos verificar con el siguiente código:

```
>_

import numpy as np
from sklearn.decomposition import PCA

# random data
X = np.random.rand(2_000, 1_000)  # (n, d)

# fit PCA
pca = PCA()
pca.fit(X)

# Manually compute singular values of:
# 1) covariance matrix.
svals_cov = np.linalg.svd(np.cov(X, rowvar=False), compute_uv=False)
# 2) *centered* data matrix.
svals_data = np.linalg.svd(X - X.mean(axis=0, keepdims=True), compute_uv=False)

# Since Cov(X) = 1/(n-1) X_cent^T X_cent and X_cent = U S V^T, we have:
#     Cov(X) = 1/(n-1) V S^2 V^T,          i.e.
#     svals_cov = svals_data^2 / (n-1):
print(np.allclose(svals_cov, svals_data**2 / (X.shape[0] - 1))) # prints True
# or equivalently, svals_data = sqrt(svals_cov * (n-1)):
print(np.allclose(svals_data, np.sqrt(svals_cov * (X.shape[0] - 1))))
# prints True

# Scikit-learn PCA's singular_values_ represents svals_data:
```

```python
print(np.allclose(pca.singular_values_, svals_data)) # prints True
# in other words: pca.singular_values_ = svals_data = sqrt(svals_cov * (n-1))
print(np.allclose(pca.singular_values_, np.sqrt(svals_cov * (X.shape[0] - 1))))
# prints True
```