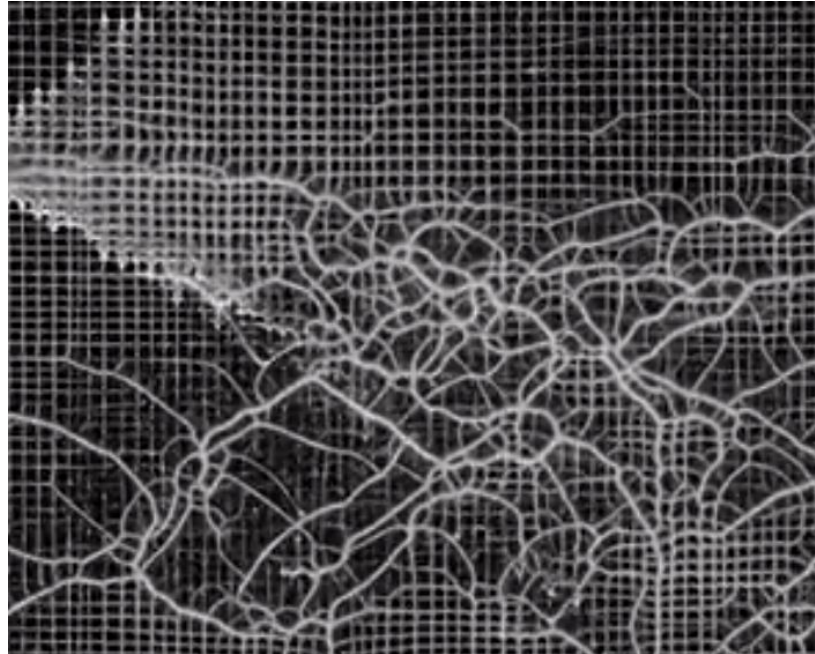


clusterAI 2020
ciencia de datos en ingeniería
industrial
UTN BA
curso I5521

clase_06: Reducción de la dimensionalidad

Docente: Martin Palazzo

AI ART



*<https://sagejenson.com/>

agenda clase05

- Reducción de la dimensionalidad
 - Principal Component Analysis (PCA)
 - Kernel PCA
- Practica 1 python
- Practica 2 python

Alta dimension

A la hora de registrar datos de un sistema se suelen realizar múltiples mediciones. Estas mediciones, luego de un pre-procesamiento, pueden ser \mathbf{d} variables/dimensiones/features que definen el espacio \mathbf{X} de nuestro problema de machine learning. Es decir que nuestro problema inicial esta descrito en un espacio de dimensión \mathbf{d} .

- Existe la hipótesis de que la variabilidad intrínseca de los datos en cuestión vive en un sub-espacio \mathbf{Z} (espacio latente) de dimensión \mathbf{p} dentro de \mathbf{X} donde $\mathbf{p} < \mathbf{d}$. Es decir que la dimensión intrínseca de los datos es menor que la original.
- Por otro lado, a mayor cantidad de dimensiones más complejo será el modelo que tenga que ser entrenado, mas parametros \mathbf{w} tendra que aprender y si la cantidad de instancias/muestras que se dispone es baja entonces estará amenazado por la maldición de la dimensionalidad (the curse of dimensionality) cuando $n/d < 1$.
- Por estas razones trataremos de encontrar (aprender) un subespacio \mathbf{z} que retenga la mayor cantidad de información, estructura y variabilidad original de los datos.
- En definitiva: si el desempeño de modelos de machine learning se mantiene o hasta mejora con menor dimensión entonces se sostiene la hipótesis de que muchas de las dimensiones originales están compuestas de información y también de ruido.

Tipo de reducción de dimensión

Dependiendo de si se usan o no las etiquetas \mathbf{y} de las muestras la reducción sera:

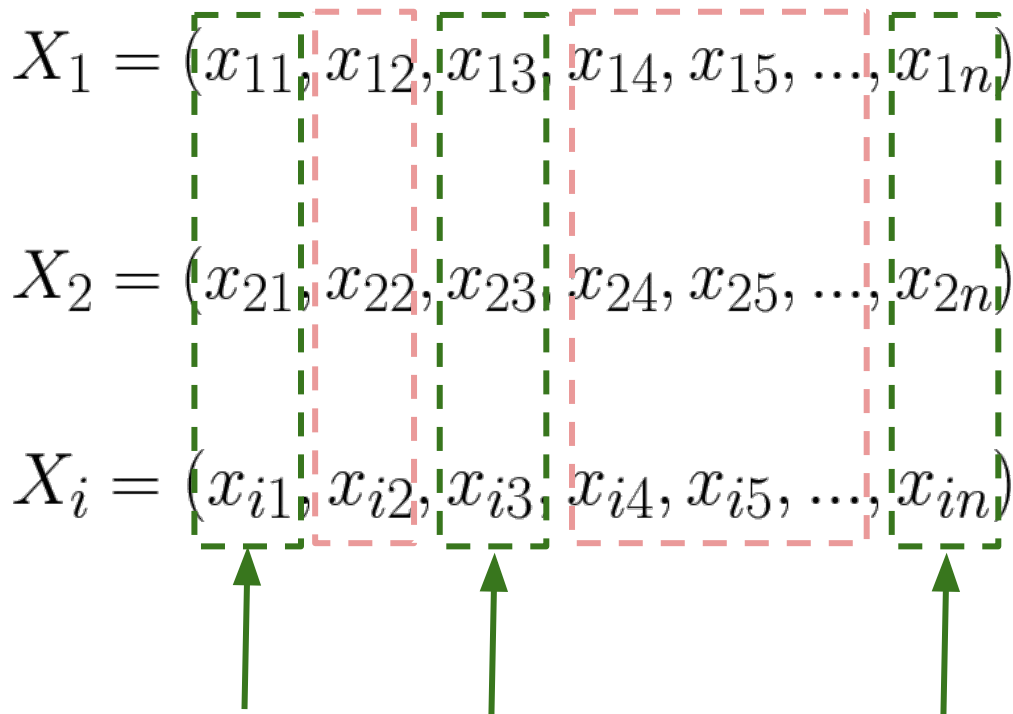
- **Reducción de la dimensionalidad supervisada:** Se busca que el sub-espacio \mathbf{Z} obtenido luego de la reducción contenga información respecto de las etiquetas \mathbf{y} para mejorar una tarea de aprendizaje supervisado “aguas abajo” (downstream task) a realizarse en \mathbf{Z} .
- **Reducción de la dimensionalidad no-supervisada:** Se busca que el sub-espacio \mathbf{Z} preserve la estructura de los datos entendiendo por estructura a la similaridad entre muestras independientemente de cualquier etiqueta \mathbf{y} . La reducción no supervisada busca mejorar tareas de aprendizaje aguas abajo sean supervisadas o no supervisadas. En muchos casos la reducción no supervisada puede filtrar el ruido de los datos (denoising).

Reducción de la dimensión

Existen dos estrategias para reducir la dimensión de los datos:

- **Selección de variables (feature selection)** -> se selecciona un sub-conjunto de variables p del conjunto inicial de d variables. Las p variables seleccionadas son de la misma naturaleza que las variables originales, por esta razón es posible interpretar el sistema.
- **Extracción de características (feature extraction)** -> mediante la combinación lineal o no lineal de las d variables iniciales se aprenderán/extraerán p nuevas variables para describir a las mismas muestras/samples.

Dim Reduction: Feature Selection



La selección de variables busca descartar las features que no son necesarias y conservar un número reducido de features.

En este caso no se realiza conversión o transformación de features. Las features seleccionadas siguen siendo representativas del contexto de los datos.

Existen distintas estrategias para seleccionar features:

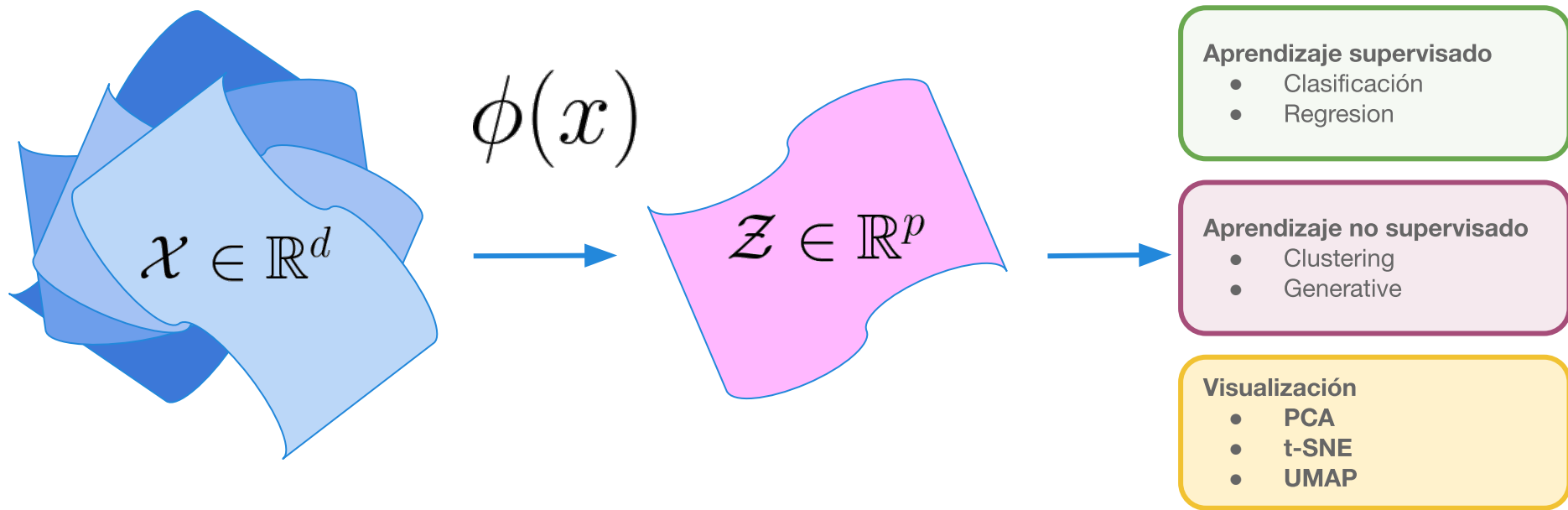
- Variance Threshold
- Recursive Feature Elimination
- Lasso
- Minimum Redundancy Maximum Relevance

Dim. Reduction: Feature extraction

$$\phi(x) = z \quad \begin{array}{l} \mathcal{X} \in \mathbb{R}^d \\ \mathcal{Z} \in \mathbb{R}^p \\ p < d \end{array}$$

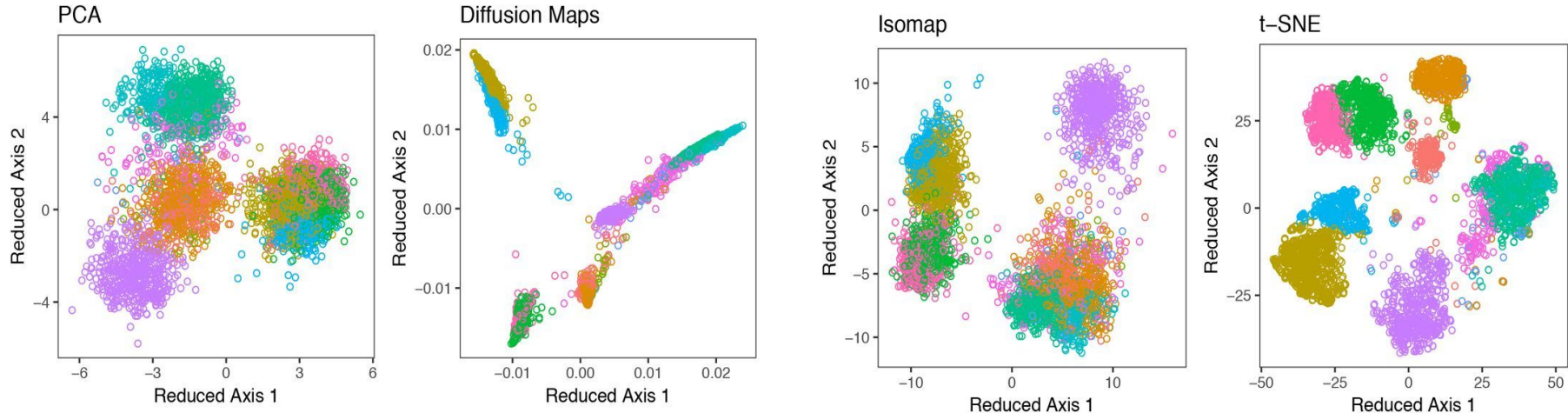
Se busca aprender una función de transformación **phi** que proyecta los datos de **X** al sub-espacio **Z** de menor dimensión. En este curso todas las transformaciones que estudiamos son no-supervisadas aunque existen supervisadas.

Dim. Reduction: Feature extraction



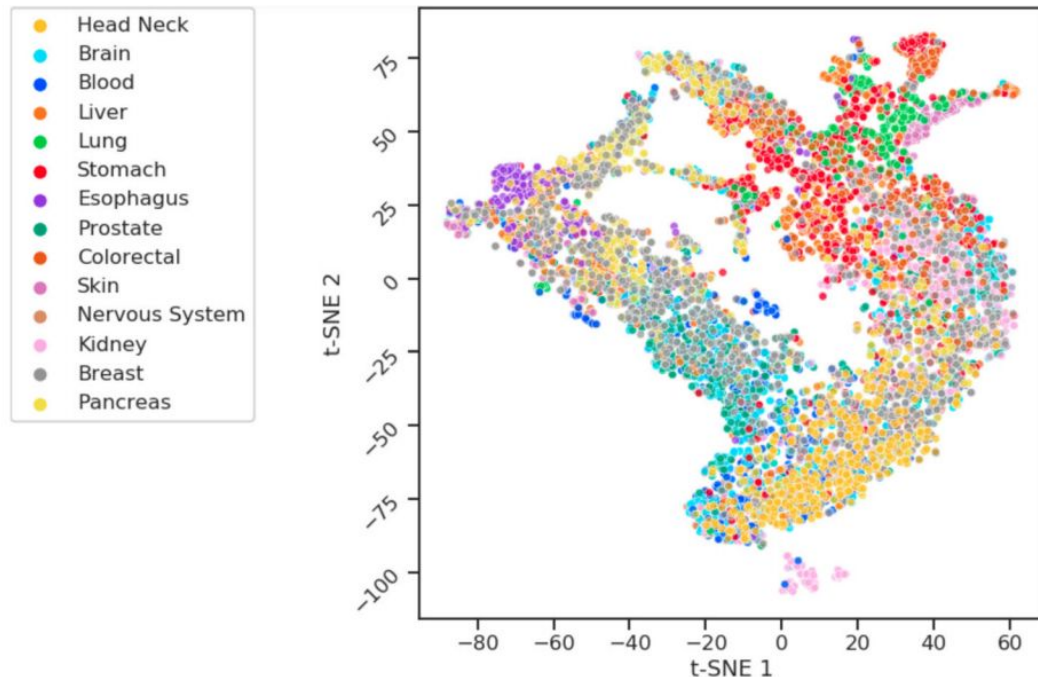
Se busca aprender una función de transformación **phi** que proyecta los datos de **X** al sub-espacio **Z** de menor dimensión. En este curso todas las transformaciones que estudiamos son no-supervisadas aunque existen supervisadas.

Dim. Red. for visualization

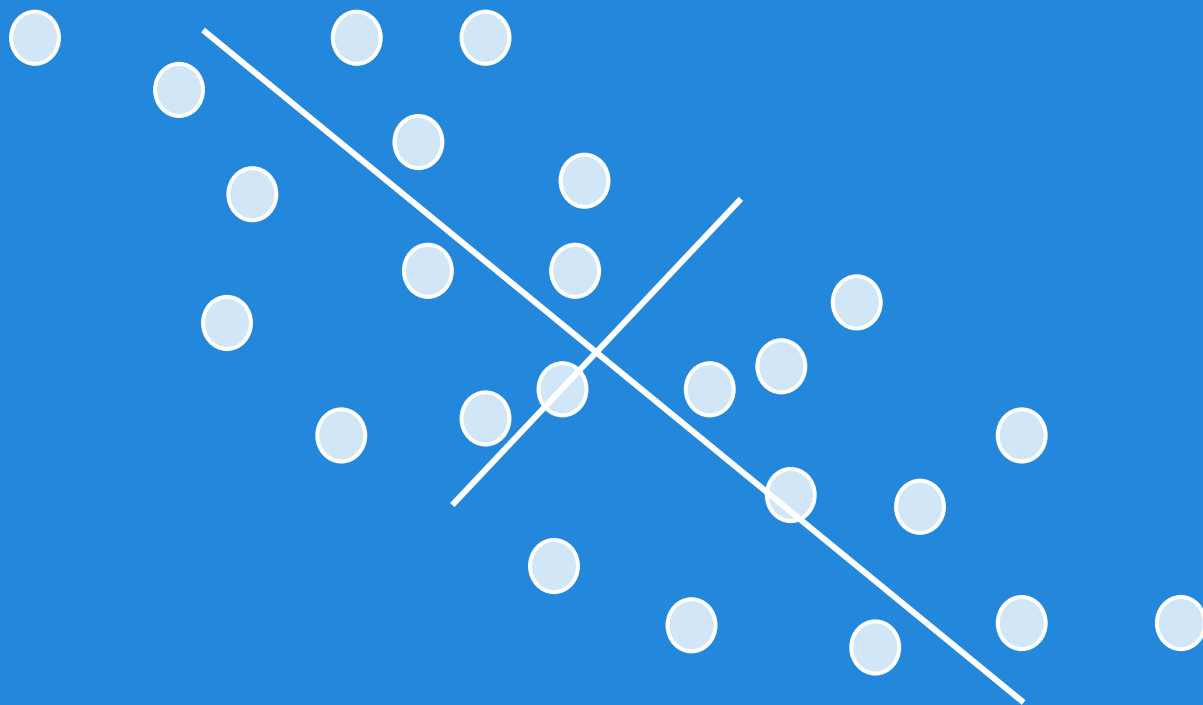


Los métodos de reducción de dimensionalidad no solo se usan para mejorar una tarea de aprendizaje supervisado o no supervisado “downstream”, además se pueden usar para visualizar en dos dimensiones datos de alta dimensión.

Dimension Reduction

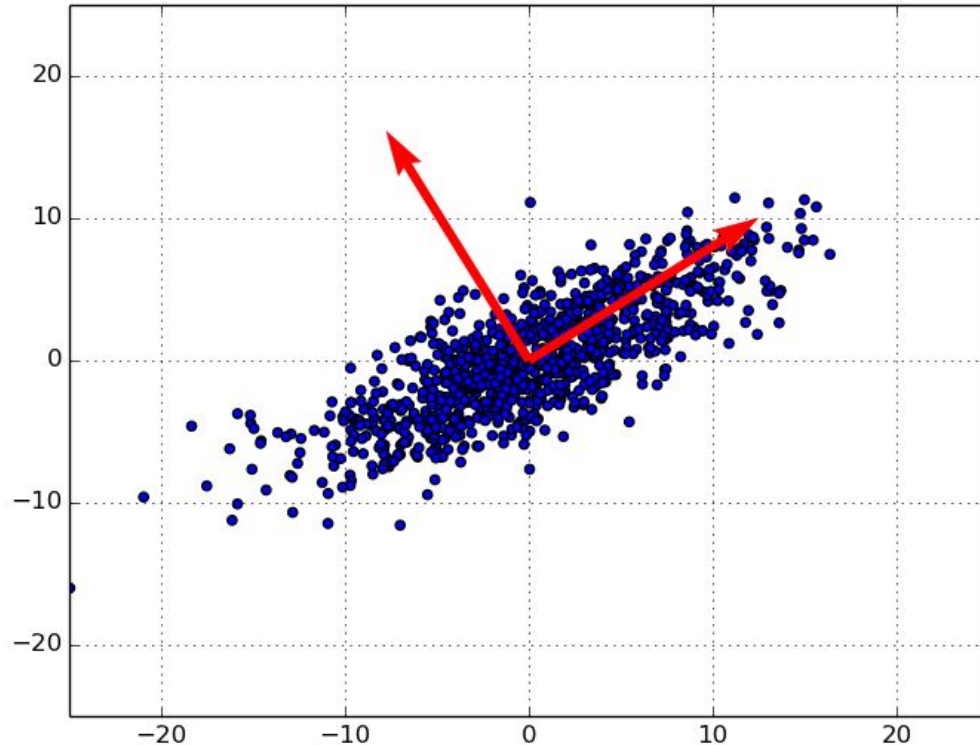


Visualización de 9000 perfiles tumorales caracterizados por mutaciones somáticas de pacientes y etiquetados en 16 tipos de cáncer. La dimensión original supera las 10.000 features. La reducción se hizo por el método de Autoencoder (redes neuronales) y la visualización del espacio latente con el método t-SNE.



Análisis de componentes principales - Principal Component Analysis (PCA)

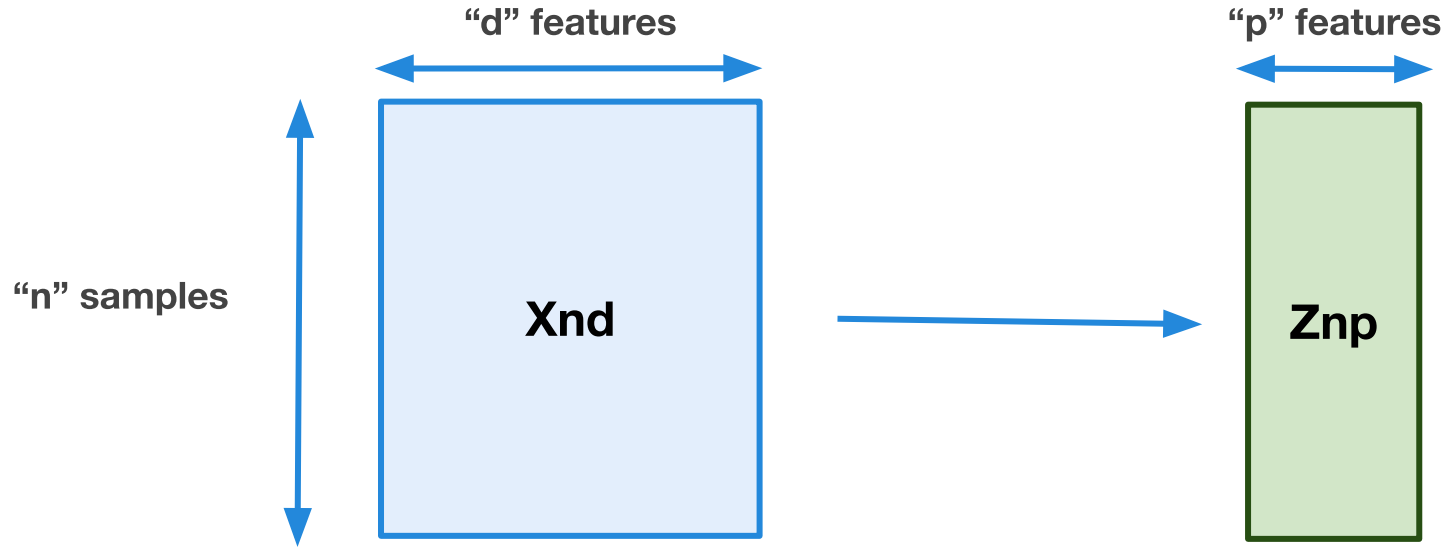
Principal Component Analysis



Principal Component Analysis (PCA)

- Es un método popular utilizado para reducir la cantidad de dimensiones(features), creando nuevas features llamadas “**Componentes Principales**” mediante la descomposición espectral (obtener auto-valores y auto-vectores). Lo que busca el PCA es proyectar en una dimensión menor nuestros datos tratando de retener la mayor cantidad de información (variabilidad) posible.
- Las nuevas dimensiones no representan nada del entorno real de los datos, más bien son **combinaciones lineales** de las features originales creadas con el fin de representar la mayor cantidad de variación de los datos. La PC01 es la componente que más variación explica, la PC02 es la segunda componente que más variación explica y así sucesivamente.
- El PCA puede ser útil para visualizar data de alta dimensionalidad en solo 2 dimensiones utilizando las primeras dos componentes.

Principal Component Analysis (PCA)



Por PCA transformaremos nuestra matriz de datos \mathbf{X}_{nd} a una matriz de datos \mathbf{Z}_{np} de menor dimension. Reducir la dimensionalidad implica encontrar un nuevo espacio latente que explica mejor mis datos y por ende puede ayudar en el entrenamiento de un modelo de aprendizaje supervisado.

Principal Component Analysis (PCA)

Uno de los objetivos es reducir la dimensionalidad de un dataset d -dimensional a otro p -dimensional donde $d > p$. Esto lo hacemos para filtrar el ruido, mejorar la eficiencia computacional y poder visualizar (en 2 o 3 dimensiones) tratando de retener la mayor cantidad de información posible.

La pregunta que surge es: **¿cuánto debe valer “ p ” para realizar una representación aceptable de los datos? ¿cómo calculo las p dimensiones nuevas?**

PCA: algoritmo

1. Estandarizar / Auto-escalar los datos **X** (media = 0, stdev = 1)
2. Computar la matriz **Sigma** de covarianza desde los datos **X**
3. Realizar descomposición en autovalores (eigen-values) y autovectores (eigen-vectors) de **Sigma**.
4. Ordenar los autovalores de mayor a menor.
5. Construir una nueva matriz **Z** utilizando los primeros “p” autovectores asociados a los p auto-valores mas grandes.
6. El nuevo dataset estará proyectado en los “p” autovectores, llamados componentes principales.

PCA: matriz de covarianza

La co-varianza entre la variable **j** y **k** a lo largo de todas las **n** muestras se define como:

$$\sigma_{jk} = \frac{1}{n-1} \sum_1^N (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$$

Y representa en un escalar como co-varían los pares de variables en cuestión. La extensión de covarianza a un dataset **X**nd sera

$$\Sigma = ((X - \bar{x})^T (X - \bar{x})) \quad \Sigma = \begin{bmatrix} \sigma_{11} & \dots & \sigma_{1d} \\ \dots & \dots & \dots \\ \sigma_{d1} & \dots & \sigma_{dd} \end{bmatrix}$$

La matriz de covarianza contempla todos los pares de variable. Similar a la matriz de correlación, la correlacion, con la diferencia de que la covarianza no esta limitada entre 0 y 1.

PCA: Eigen-decomposition

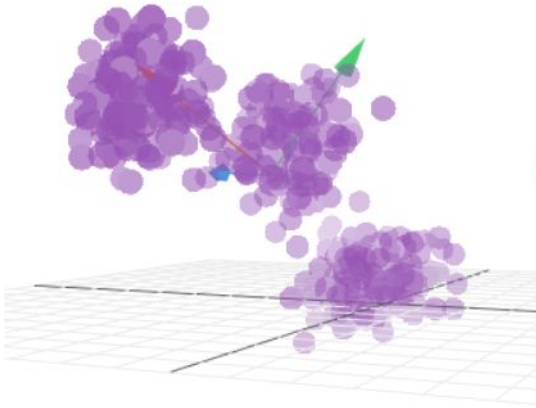
La matriz de covarianza **Sigma** puede ser decompuesta en una matriz de auto-vectores V (eigenvectors) y auto-valores lambda (eigen-decomposition).

$$\Sigma = v \lambda v^{-1} \rightarrow \beta_{dp} \rightarrow X.\beta = z$$

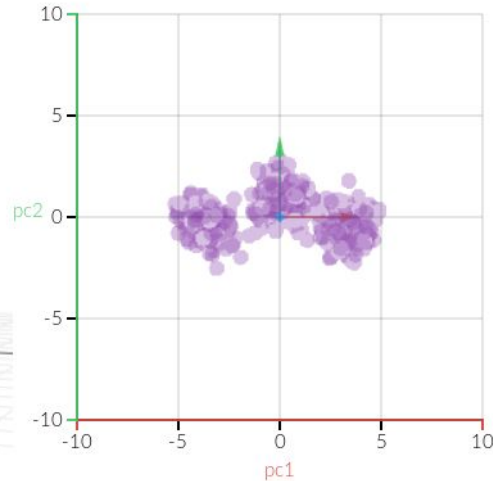
Cada autovector estará asociado a un autovalor. El autovalor indicará la cantidad de variabilidad que explica cada auto-vector de los datos. Buscaremos retener los auto-vectores asociados a los auto-valores mas grandes. Con los p autovectores que más variabilidad explican se construye la matriz de proyección al sub-espacio **Z**.

Principal Component Analysis (PCA)

Antes del PCA



Después del PCA



Las componentes principales después de computar los **autovectores** y **autovalores** de la matriz de **covarianza** de nuestros datos.

Los autovectores nos indicarán cómo proyectar los datos X sobre las componentes principales y los autovalores indicarán el nivel de importancia (variabilidad) de cada componente principal.

Principal Component Analysis (PCA)



[Home](#) [Installation](#) [Documentation](#) [Examples](#)

Google Custom Search



Previous

[sklearn.deco
m_...](#)

Next

[sklearn.deco
m_...](#)

Up

[API
Reference](#)

scikit-learn v0.21.dev0

[Other versions](#)

Please [cite us](#) if you
use the software.

[sklearn.decomposition.PCA](#)
Examples using
[sklearn.decomposition.PCA](#)


sklearn.decomposition.PCA

```
class sklearn.decomposition. PCA (n_components=None, copy=True, whiten=False, svd_solver='auto', tol=0.0,  
iterated_power='auto', random_state=None) \[source\]
```

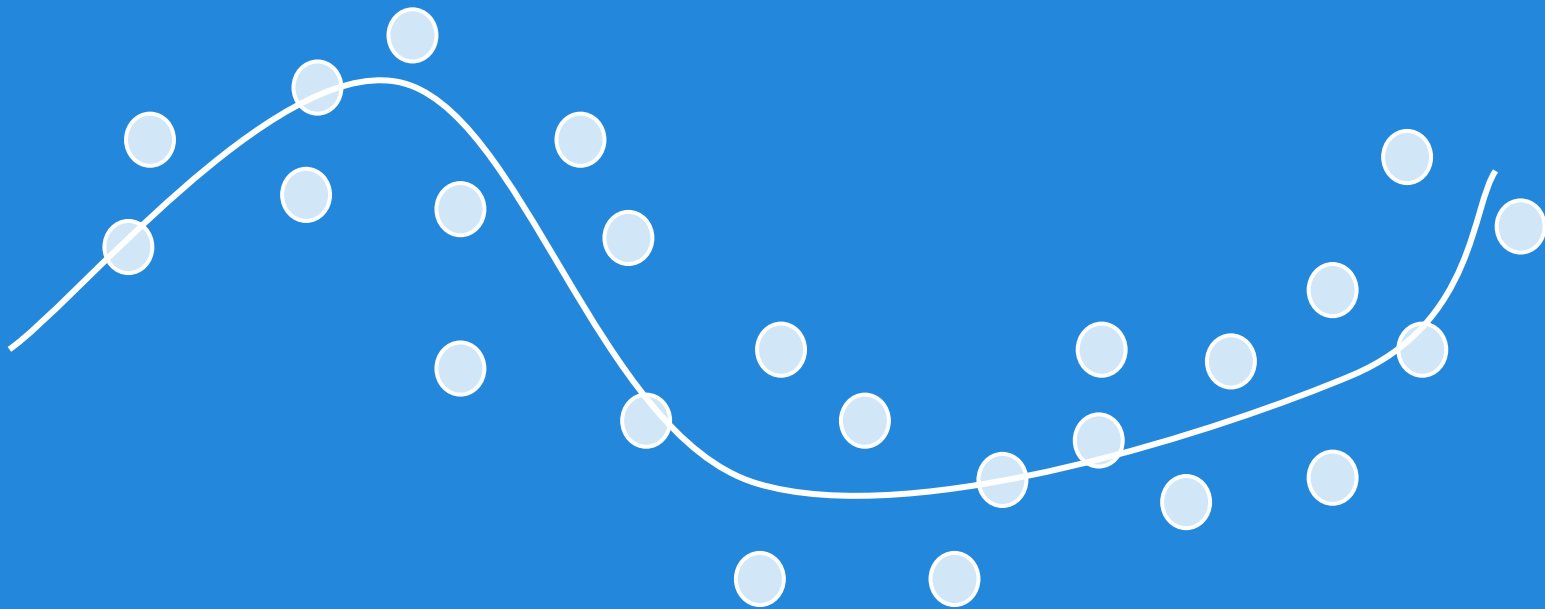
Principal component analysis (PCA)

Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.

PCA



```
from sklearn.decomposition import PCA
# Dataset (ojo no hay labels!!!)
xtrain = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3,
2]])
# Definimos el PCA y cuantas componentes queremos retener
pca = PCA(n_components=2)
# Fiteamos el PCA a nuestros datos
pca.fit(xtrain)
# Podemos observar cada auto-valor ya ordenado
print(pca.explained_variance_ratio_)
[0.9924... 0.0075...]
# Obtenemos los datos proyectados en las componentes seleccionadas
xpca_train = pca.fit_transform(xtrain)
```



Kernel PCA

kernelPCA: reducción no-lineal

Funciones de similaridad

$$k_{\text{rbf}}(x_i, x_j) = \exp \left(-\gamma \| \mathbf{x}_i, \mathbf{x}_j \|^2 \right)$$

$$k_{\text{pol}}(x_i, x_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + C)^d$$

Matriz de similaridad

	1	2	...	N
1	$\kappa(\mathbf{x}_1, \mathbf{x}_1)$	$\kappa(\mathbf{x}_1, \mathbf{x}_2)$	\cdots	$\kappa(\mathbf{x}_1, \mathbf{x}_N)$
2	$\kappa(\mathbf{x}_2, \mathbf{x}_1)$	$\kappa(\mathbf{x}_2, \mathbf{x}_2)$	\cdots	$\kappa(\mathbf{x}_2, \mathbf{x}_N)$
\vdots	\vdots	\vdots	\cdots	\vdots
N	$\kappa(\mathbf{x}_N, \mathbf{x}_1)$	$\kappa(\mathbf{x}_N, \mathbf{x}_2)$	\cdots	$\kappa(\mathbf{x}_N, \mathbf{x}_N)$

Con un kernel no-lineal vamos a calcular similaridad entre pares de muestras i-j. La similaridad devuelve un número entre 0 y 1. Con el dataset podemos armar la matriz de similaridad **K**.

kernelPCA: reducción no-lineal

$$K_{(nn)} = nU^{-1}\Lambda U \longrightarrow Z_{(np)} = KU^*$$

Se realizará eigen-decomposition en la matriz \mathbf{K} para obtener auto-valores \mathbf{U} y auto-vectores $\mathbf{\Lambda}$. Se proyectarán los datos utilizando de matriz \mathbf{U} de los primeros \mathbf{p} autovectores quedando la proyección U^* .

Dim. Red. nonlinear: kernel PCA

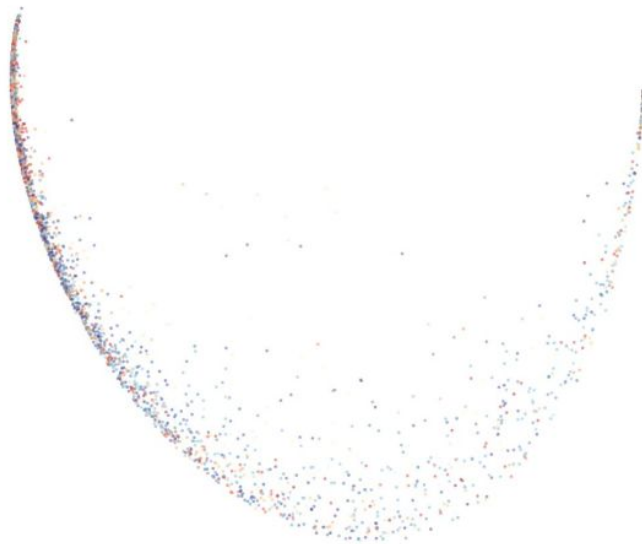


Figure: A Kernel PCA visualization