Extracción de características para reducir la Dimensionalidad

¿Qué aprenderemos hoy?

- **Análisis de Componentes Principales (PCA)**

Análisis de Componentes Principales

- **Algoritmo de Aprendizaje No Supervisado.**
- **Transforma o proyecta (linealmente) los datos en un nuevo espacio de características.**

En el contexto de la reducción de la dimensionalidad, la extracción de características puede entenderse como un enfoque de compresión de datos con el objetivo de mantener la mayor parte de la información relevante.

En la práctica, la extracción de características no sólo se utiliza para mejorar el espacio de almacenamiento (en memoria) o la eficiencia computacional del algoritmo de aprendizaje, sino que también puede mejorar el rendimiento predictivo, al reducir **la maldición de la dimensionalidad**, especialmente si trabajamos con modelos no regularizados.

PCA nos ayuda a identificar patrones basados en la **correlación entre las características**. De esta manera, **PCA tiene como objetivo encontrar las direcciones de máxima varianza en los datos y los proyecta en un nuevo subespacio con dimensión igual o menor que el original**. Los ejes ortogonales (componentes principales) del nuevo subespacio pueden interpretarse como las direcciones de máxima varianza dada la restricción de que los nuevos ejes de características son ortogonales entre sí.

In [ ]:
```
from IPython.display import Image

Image(filename=r'Imagenes_clase_8/8_1.png', width=500)
```

Al utilizar PCA para reducir la dimensión de los datos, construimos una matriz de transformación W de dimensión $d \times k$, que nos permite mapear un vector de una muestra x en un nuevo subespacio de características k-dimensional, con $k \leq d$.

$$\boldsymbol{x} = [x_1, x_2, \cdots, x_d], \;\; \boldsymbol{x} \in \mathbb{R}^d$$

$$\boldsymbol{z} = \boldsymbol{x}\boldsymbol{W}, \;\; \boldsymbol{W} \in \mathbb{R}^{d \times k}$$

$$\boldsymbol{z} = [z_1, z_2, \cdots, z_k], \;\; \boldsymbol{z} \in \mathbb{R}^k$$

Algoritmo PCA:

1. Normalizar el conjunto de datos X, $N \times d$-dimensional.
2. Construir la matriz de covarianza.
3. Descomponer la matriz de covarianza en sus vectores y valores propios.
4. Ordenar los valores propios por orden decreciente para clasificar los correspondientes vectores propios.
5. Seleccionar k vectores propios, donde estos corresponden a los vectores asociados a los k mayores valores propios, siendo k la dimensión del nuevo subespacio de características ($k \leq d$).
6. Construir una matriz de proyección W a partir de los k primeros vectores propios.
7. Transformar el conjunto de datos X de entrada utilizando la matriz de proyección W, ie, calcular $z = xW$ para todo $x \in X$.

In [ ]:
```
import pandas as pd

df_wine = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data',header=None)

df_wine.columns = ['Class label', 'Alcohol', 'Malic acid', 'Ash',
                   'Alcalinity of ash', 'Magnesium', 'Total phenols',
                   'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
                   'Color intensity', 'Hue',
                   'OD280/OD315 of diluted wines', 'Proline']

df_wine
```

In [ ]:
```
df_wine['Class label'].value_counts()
```

In [5]:
```
from sklearn.model_selection import train_test_split

X, y = df_wine.iloc[:, 1:].values, df_wine.iloc[:, 0].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y, random_state=0)
```

Paso 1: Estandarizar

In [6]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

Paso 2: Matriz de covarianza

La Matriz de covarianza es de dimensión $d \times d$. Almacena las covarianzas por pares entre las diferentes características. Por ejemplo, la covarianza entre dos características x_j y x_k puede calcularse mediante la siguiente ecuación:

$$\sigma_{jk} = \frac{1}{N}\sum_{i=1}^{N}\left(x_j^{(i)} - \mu_j\right)\left(x_k^{(i)} - \mu_k\right)$$

μ_j y μ_k son las medias muestrales de las características j y k, respectivamente.

Una covarianza positiva entre dos características indica que las características aumentan o disminuyen juntas, mientras que una covarianza negativa indica que las características varían en direcciones opuestas.

Matriz de covarianza de tres características:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 \end{bmatrix}$$

Los vectores propios de la matriz de covarianza representan los componentes principales (las direcciones de máxima varianza), mientras que los valores propios correspondientes representan su magnitud.

Recordatorio:

Si v es un vector propio, entonces $\Sigma v = \lambda v$, λ es un escalar; el valor propio.

Paso 3: Descomponer matriz de covarianza

In [ ]:
```
import numpy as np

print(f'Dimensiones de X_train_std: {X_train_std.shape}')
```

In [ ]:
```
cov_mat = np.cov(X_train_std.T)
cov_mat
```

In [ ]:
```
cov_mat.shape
```

In [ ]:
```
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)

print('\nValores Propios: \n%s' % eigen_vals)
```

Varianza total y explicada:

Como queremos reducir la dimensionalidad de nuestro conjunto de datos, comprimiéndolo en un nuevo subespacio de características, sólo seleccionaremos el subconjunto de los vectores propios (componentes principales) que contiene la mayor parte de la información (varianza). Los valores propios definen la magnitud de los vectores propios, por lo que tenemos que ordenar los valores propios por magnitud decreciente.

La proporción de varianza explicada de un valor propio λ_j es simplemente: $\frac{\lambda_j}{\sum_{j=1}^{d}\lambda_j}$

Paso 4: Ordenar valores propios

In [ ]:
```
tot = sum(eigen_vals)
var_exp = [(i / tot) for i in sorted(eigen_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print(np.array(var_exp))
print(cum_var_exp)
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.bar(range(1, 14), var_exp, alpha=0.5, align='center',
        label='varianza explicada (individual)')
plt.step(range(1, 14), cum_var_exp, where='mid',
         label='varianza explicada acumulada')
plt.ylabel('Varianza Explicada (Ratio)')
plt.xlabel('Indice de las Componentes Principales')
plt.legend(loc='best')
plt.tight_layout()
# plt.savefig('images/05_02.png', dpi=300)
plt.show()
```

Transformación de las características

Paso 5: Seleccionar k vectores propios, que corresponden a los k mayores valores propios.

Paso 6: Construir una matriz de proyección W a partir de los k vectores propios seleccionados.

Paso 7: Transformar el conjunto de datos X de entrada (dimesión $N \times d$) utilizando la matriz de proyección W y así obtener el nuevo subespacio de características k-dimensional y un conjunto de datos X_0 de dimensión $N \times k$, con $k \leq d$.

In [ ]:
```
eigen_pairs = [(np.abs(eigen_vals[i]), eigen_vecs[:, i]) for i in range(len(eigen_vals))]
eigen_pairs
```

In [ ]:
```
eigen_pairs.sort(key=lambda k: k[0], reverse=True)
eigen_pairs
```

In [ ]:
```
print(eigen_pairs[0][1])
```

In [ ]:
```
eigen_pairs[0][1][:, np.newaxis]
```

In [ ]:
```
w = np.hstack((eigen_pairs[0][1][:, np.newaxis], eigen_pairs[1][1][:, np.newaxis]))

print('Matriz W:\n', w)
print('Dim W:\n', w.shape)
```

In [ ]:
```
X_train_std[0]
```

In [ ]:
```
X_train_std[0].dot(w)
```

In [ ]:
```
X_train_std[15].dot(w)
```

Ahora, $X_0 = XW$

In [ ]:
```
X_train_pca = X_train_std.dot(w)
colors = ['r', 'b', 'g']
markers = ['s', 'x', 'o']

for l, c, m in zip(np.unique(y_train), colors, markers):
    plt.scatter(X_train_pca[y_train == l, 0],X_train_pca[y_train == l, 1],c=c, label=l, marker=m)

plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend(loc='lower right')
plt.tight_layout()
# plt.savefig('images/05_03.png', dpi=300)
plt.show()
```

PCA con Scikit-Learn

- Documentación

In [ ]:
```
from sklearn.decomposition import PCA

pca = PCA()
X_train_pca = pca.fit_transform(X_train_std)
pca.explained_variance_ratio_
```

In [ ]:
```
plt.bar(range(1, 14), pca.explained_variance_ratio_, alpha=0.5, align='center')
plt.step(range(1, 14), np.cumsum(pca.explained_variance_ratio_), where='mid')
plt.ylabel('Varianza Explicada (Ratio)')
plt.xlabel('Indice de las Componentes Principales')
plt.show()
```

In [ ]:
```
X_train_pca.shape
```

¿Cuánto contribuyen las características a las Componenetes Principales?

In [ ]:
```
pca.components_.shape
pca.explained_variance_
```

In [ ]:
```
pca.components_.T * np.sqrt(pca.explained_variance_)
```

In [ ]:
```
factor_loadings = pca.components_.T * np.sqrt(pca.explained_variance_)

fig, ax = plt.subplots()

ax.bar(range(13), factor_loadings[:, 0], align='center')
ax.set_ylabel('Factor de cargas para PC 1')
ax.set_xticks(range(13))
ax.set_xticklabels(df_wine.columns[1:], rotation=90)

plt.ylim([-1, 1])
plt.tight_layout()
#plt.savefig('figures/05_05_03.png', dpi=300)
plt.show()
```

In [ ]:
```
fig, ax = plt.subplots()

ax.bar(range(13), factor_loadings[:, 1], align='center')
ax.set_ylabel('Factor de cargas para PC 2')
ax.set_xticks(range(13))
ax.set_xticklabels(df_wine.columns[1:], rotation=90)

plt.ylim([-1, 1])
plt.tight_layout()
#plt.savefig('figures/05_05_03.png', dpi=300)
plt.show()
```

Ejercicios

Considere la base de datos Bonus del Control 2.

- Seleccione todas las variables numéricas, restaurando las que están dañadas.
- Elimine todas los valores extremos y complete los faltantes usando el promedio
- Aplique el algoritmo PCA y reduzca los datos numéricos a dimension 3.
- Describa el aporte de cada variable a las tres componentes principales.