Seleccionando características significativas

Si se observa que un modelo tiene mejor rendimiento con el conjunto de entrenamiento que con el de pruebas, es un indicador de *sobreajuste*; esto significa que el modelo adecúa sus parámetros a las observaciones de los datos de entrenamiento, pero no generaliza bien a nuevos datos.

Entre las posibles soluciones al sobreajuste, se tiene la reducción de la dimensión de los datos.

Selección secuencial de características

Los algoritmos de selección secuencial son una familia de algoritmos voraces (*greedy*) usados para reducir la dimensión d de características a un subespacio de dimensión k que cumple $k < d$. El objetivo es seleccionar automáticamente un subconjunto de características que sean las más relevantes para el problema; con esto se puede mejorar la eficiencia o reducir errores de generalización del modelo al eliminar características irrevantes o ruido.

Un algoritmo de selección común es el llamado ***Sequential Backward Selection*** (***SBS***) cuyo objetivo es reducir la dimensión del espacio de características con un mínimo de pérdidas en el rendimiento del modelo mejorando la eficiencia computacional. Además, SBS puede mejorar también el poder predictivo de un modelo si está sobreajustado.

La idea detrás de SBS es simple: eliminar secuencialmente características de un conjunto hasta que se tenga el número deseado de ellas. Para determinar qué columna eliminar se debe establecer una función criterio $\mathcal{J}$; uno muy usado es simplemente comparar el rendimiento del modelo antes y después de eliminar una columna; la característica a eliminar será aquella que cause el menor descenso en el rendimiento. El algoritmo puede describirse en cuatro pasos:

Algoritmo 0.1 *Sequential Backward Selection*

1. Inicializar el algoritmo con $k = d$, d es la dimensión del espacio original de características $\mathbf{X}_d$
2. Determinar la característica x^- que maximiza el criterio: $x^- = \arg\max\left(\mathcal{J}\left(X_k - x\right)\right)$, donde $x \in X_k$
3. Eliminar la característica x^- del conjunto: $X_{k-1} = X_k - x^-; k = k - 1$
4. Si k es mayor que el número deseado de características, repetir desde el paso 2; de otro modo, terminar la ejecución

---

Reducción de dimensión no supervisada: Análisis de Componentes Principales (PCA)

Otra forma de reducir la cantidad de datos a procesar, es la compresión de datos, dado que nos permite almacenar y analizar gran catidad de datos producidos y recolectados por medio tecnológicos. Mientras que SBS es un algoritmo de *selección*, PCA (***Principal Component Analysis***) es un algoritmo de *extracción* de características. La diferencia principal es que en un algoritmo de selección, se conservan las características originales y en uno de extracción, los datos se transforman o proyectan en un nuevo espacio de características.

PCA ayuda a identificar patrones en los datos basado en la correlación entre las características; es decir, intenta encontrar las direcciones de máxima varianza en datos con muchas dimensiones y

las proyecta en un nuevo subespacio con igual o menos dimensiones que el original. El algoritmo se compone de los siguientes pasos:

Algoritmo 0.2 *Principal Component Analysis*

1. Estandarizar el conjunto de datos de dimensión d
2. Obtener la matriz de covarianza
3. Descomponer la matriz de covarianza en sus eigenvalores y eigenvectores
4. Ordenar los eigenvalores de manera decreciente de acuerdo a sus correspondientes eigenvectores
5. Seleccionar los k eigenvectores que corresponden con los k mayores eigenvalores; k es la dimensión de nuevo subespacio de características ($k < d$)
6. Construir una matriz de proyección $\mathbf{W}$ con los primeros k eigenvectores
7. Transformar el conjunto de datos de entrada $\mathbf{X}$ de dimensión d utilizando la matriz de proyección $\mathbf{W}$ para obtener el nuevo subespacio de características de dimensión k

Implementación de PCA desde cero:

```
# Clase
```

Usando PCA de *scikit-learn*:

```
# Clase
```

Reducción de dimensión supervisada: Análisis Lineal Discriminante (LDA)

Linear Discriminant Analysis (***LDA***) es una técnica de extracción de características que puede usarse para incrementar la eficiencia computacional y reducir los sobreajustes. Formulado inicialmente por Ronald A. Fischer (https://bit.ly/3f9e6KF) en 1936 con el conjunto de datos de flores iris para problemas de clasificación de dos clases. En 1948 C. Radhakrishna Rao (https://bit.ly/2VX3Hub) lo generalizó para problemas multiclase bajo el supuesto de covarianzas de clase iguales y clases con distribuciones normales.

En general, los conceptos detrás de LDA son muy similares a PCA: mientras PCA busca las componentes ortogonales de varianza mínima, el objetivo de LDA es encontrar un subespacio de características que optimice la separabilidad de clases.

El algoritmo se compone de los siguientes pasos:

Algoritmo 0.3 *Linear Discriminant Analysis*

1. Estandarizar el conjunto de datos de dimensión d
2. Para cada clase, calcular su vector de medias (de dimensión d)
3. Obtener la matriz de dispersión (*scatter matrix*) entre clases S_B y la matriz de dispersión de la propia clase S_W
4. Determinar los eigenvalores y eigenvectores correspondientes a la matriz $S_W^{-1}S_B$
5. Ordenar los eigenvalores de manera decreciente de acuerdo a sus correspondientes eigenvectores
6. Seleccionar los k eigenvectores que corresponden con los k mayores eigenvalores para construir una $d \times k-$dimensional matriz de proyección $\mathbf{W}$; los eigenvectores son las columnas de dicha matriz
7. Proyectar las muestras sobre el nuevosubespacio de características utilizando la matriz de proyección $\mathbf{W}$

(PARA EL SÁBADO)

Reducción no lineal de dimensión: Análisis de la Componente Principal por Núcleos (KPCA)

Algunos algoritmos de aprendizaje automático suponen separabilidad lineal de los datos de entrada; por ejemplo, el perceptrón requiere separabilidad perfecta para garantizar convergencia. Por otro lado, existen algoritmos que suponen que la falta de separabilidad lineal se debe a ruido; por ejemplo, la regresión logística.

Sin embargo, si tenemos un problema no lineal, muy comunes en aplicaciones reales, las técnicas de transformación lineal para reducir la dimensión (PCA, LDA) pueden no ser una buena opción. En esta sección se presenta una versión *kernelizada* (por núcleos) de PCA: ***Kernel Principal Component Analysis*** (***KPCA***) para transformar datos que no son linealmente separables en un subespacio nuevo de menor dimensión que sea adecuado para clasificadores lineales.

Funciones de núcleos

Cuando tenemos problemas no lineales, se pueden abordar proyectándolos en un espacio de características de dimensión superior en el que las clases son linealmente separables. Para transformar las muestras $\mathbf{x} \in \mathbb{R}^d$ sobre un espacio superior $k-$dimensional, debe definirse una función de mapeo ϕ:

$$\phi : \mathbb{R}^d \to \mathbb{R}^k;\ (k \gg d)$$

ϕ puede verse como una función que genera combinaciones no lineales de las características originales del conjunto de datos $d-$dimensional original sobre un espacio de características $k-$dimensional. Por ejemplo, para un vector de dos dimensiones $\mathbf{x} \in \mathbb{R}^2$ un mapeo potencial al espacio de 3 dimensiones $\mathbb{R}^3$ puede ser:

$$\mathbf{x} = [x_1, x_2]^T$$

$$\downarrow$$
$$\phi$$
$$\downarrow$$

$$\mathbf{z} = \left[x_1^2, \sqrt{2x_1x_2}, x_2^2\right]^T$$

De esta forma, se puede utilizar PCA en un espacio de orden superior para después proyectarlo sobre un espacio de dimensión menor en el que las muestras puedan separarse con un clasificador lineal.

Para implementar un KPCA con *función de base radial* (***Radial Basis Function*** **RBF** o Gaussiano) como kernel, se realizan los siguientes pasos:

Algoritmo 0.4 *Kernel PCA*

1. Obtener la matriz de núcleos $\mathbf{K}$, donde debe calcularse: $\mathtt{k}\left(x^{(i)}, x^{(j)}\right) = \exp\left(-\gamma \left\|x^{(i)} - x^{(j)}\right\|^2\right)$; $\gamma = \dfrac{1}{2\sigma}$ para cada par de muestras:
$$\mathbf{K} = \begin{bmatrix} \mathtt{k}\left(x^{(1)}, x^{(1)}\right) & \mathtt{k}\left(x^{(1)}, x^{(2)}\right) & \cdots & \mathtt{k}\left(x^{(1)}, x^{(n)}\right) \\ \mathtt{k}\left(x^{(2)}, x^{(1)}\right) & \mathtt{k}\left(x^{(2)}, x^{(2)}\right) & \cdots & \mathtt{k}\left(x^{(2)}, x^{(n)}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \mathtt{k}\left(x^{(n)}, x^{(1)}\right) & \mathtt{k}\left(x^{(n)}, x^{(2)}\right) & \cdots & \mathtt{k}\left(x^{(n)}, x^{(n)}\right) \end{bmatrix}$$
Es decir, si tenemos 100 muestras de entrenamiento, su matriz de núcleos sería de dimensión 100×100
2. Centrar la matriz $\mathbf{K}$ utilizando: $\mathbf{K}' = \mathbf{K} - 1_n\mathbf{K} - \mathbf{K}1_n + 1_n\mathbf{K}1_n$; donde 1_n es una matriz $n \times n-$dimensional en la que todos los valores son $\dfrac{1}{n}$
3. Seleccionar los k eigenvectores de la matriz centrada que corresponden con los k mayores eigenvalores

El centrado de la matriz (paso 2) se requiere porque no es posible garantizar que el nuevo espacio esté centrado en cero. Otras funciones de núcleo comúnmente usadas son el *kernel polinomial* y el *kernel tangente hiperbólico* (sigmoide).

Implementación de RBF Kernel PCA:

```
# Clase
```