Métodos Numéricos
2^{do} Cuatrimestre 2024
Trabajo Práctico 2

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Análisis de componentes principales y procesamiento de lenguage

---

Introducción

El objetivo de este trabajo es poder clasificar textos que describen películas para identificar el género de la película. Para esto vamos a utilizar un conjunto de datos llamado Wikipedia Movie Plots [1] el cual consiste resúmenes de películas extraídos de Wikipedia. Al mismo tiempo vamos a realizar un Análisis de Componentes Principales (PCA, por las siglas en ingles) sobre este conjunto de datos y observar si varía la performance en el reconocimiento utilizando distinta cantidad de componentes.

Para procesar textos y poder construir vectores de atributos para poder alimentar un sistema de clasificación es necesario tomar ciertos pasos. Para simplificar, el dataset que vamos a usar uno que ya está procesado y tiene solo 4 géneros de películas: western, ciencia ficción, romántico y crimen. Los resúmenes de estos películas son procesados de tal forma de generar para cada película un vector de atributos X_i el cual indica la cantidad de veces que aparecen ciertos tokens. La forma de elegir estos tokens es fundamental para poder tener un buen reconocimiento. El conjunto de datos que vamos a utilizar tiene 9581 tokens, por lo que el conjunto de datos puede pensarse como una matriz $X \in \mathbb{R}^{N \times 9581}$ con N la cantidad de películas. Para estudiar si de estas 9581 dimensiones son todas importantes o hay cierta redundancia (por ejemplo, dos palabras que siempre aparecen al mismo tiempo) descartaremos ciertas dimensiones en base a si ocurren muy pocoe y luego usaremos PCA para reducir la dimensionalidad del problema.

La tarea de clasificación involucra crear un sistema que toma un elemento del conjunto de datos y genera una predicción, en este caso un valor entre 1 y 4 que representa a cada uno de los géneros. Para armar este clasificador vamos a utilizar la técnica de aprendizaje automático de k-vecinos más próximos (en ingles, *k-nearest neighbors*, KNN).

Una de las principales características de un sistema de aprendizaje automático en base a datos, es poder optimizar un objetivo a partir de un conjunto de datos de entrenamiento y poder mantener la performance en datos no vistos durante el entrenamiento. Para evaluar este escenario, se suele tomar distintos datos para entrenar y medir la performance. Si se cuenta con un solo conjunto de datos, la práctica más común es tomar una proporción de ellos mayoritaria como conjunto de entrenamiento y el resto dejarlo como conjunto de prueba. Un paso importante acá, es identificar que criterios son importantes que el conjunto de prueba cumpla y separar los datos en base a eso.

Metodología

Clasificación con k-vecinos más cercanos

En su versión más simple, este algoritmo considera a cada objeto del conjunto de entrenamiento como un punto en el espacio euclídeo m-dimensional, para el cual se conoce a qué clase corresponde (en nuestro caso, qué prenda es) para luego, dado un nuevo objeto, asociarle

[1] `https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots`

la clase del o los puntos más cercanos del conjunto de datos. La dimensión m es la cantidad atributos que describen al resumen de la películas, en nuestro caso la cantidad palabras claves o también será la cantidad de componentes principales.

Procedimiento de k vecinos más cercanos

- Se define una base de datos de entrenamiento como el conjunto $\mathcal{D} = \{x_i \ : \ i = 1, \ldots, n\}$.
- Dada una nueva película $x \in \mathbb{R}^m$, talque $x \notin \mathcal{D}$, para clasificarla simplemente se busca el subconjunto de los k vectores $\{x_i\} \subseteq \mathcal{D}$ más cercanos a x, y se le asigna la clase que posea el mayor número de repeticiones dentro de ese subconjunto, es decir, la moda.

Podemos notar que el clasificador toma como argumento el valor constante k (llamado hiperparámetro en jerga de aprendizaje automático). Los resultados de la predicción pueden variar drásticamente según el valor de k y la naturaleza de los datos. Otro factor que puede alterar los resultados es el criterio que se tome para medir la distancia entre dos objetos. Por lo tanto las predicciones pueden ser sensibles a la dimensionalidad.

Covarianza y correlación

Dados dos vectores $\mathbf{x}$ y $\mathbf{y}$ definimos la covarianza como

$$Cov(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{x} - \mu_x) \cdot (\mathbf{y} - \mu_y)}{n - 1} \tag{1}$$

Donde μ representa el valor medio del vector. Esta fórmula se puede interpretar como el producto interno de los vectores "centrados" y normalizada por la dimensión del vector menos uno.

Luego, definimos la correlación como una covarianza normalizada para que su rango sea entre -1 y 1.

$$Corr(\mathbf{x}, \mathbf{y}) = \frac{(\mathbf{x} - \mu_x) \cdot (\mathbf{y} - \mu_y)}{\sqrt{(\mathbf{x} - \mu_x) \cdot (\mathbf{x} - \mu_x)(\mathbf{y} - \mu_y) \cdot (\mathbf{y} - \mu_y)}} \tag{2}$$

Esta expresión es equivalente al coseno del ángulo entre los vectores $\cos \theta_{\mathbf{xy}}$. Alta correlación implica vectores paralelos, con correlación positiva para vectores en la misma dirección y negativa en direcciones opuestas, y correlación nula implica vectores perpendiculares. A partir de la correlación podemos definir la distancia coseno centrada como:

$$D_{\text{coseno}}(\mathbf{x}, \mathbf{y}) = 1 - Corr(\mathbf{x}, \mathbf{y}) = 1 - \frac{(\mathbf{x} - \mu_x) \cdot (\mathbf{y} - \mu_y)}{\sqrt{(\mathbf{x} - \mu_x) \cdot (\mathbf{x} - \mu_x)(\mathbf{y} - \mu_y) \cdot (\mathbf{y} - \mu_y)}}$$

Si se consideran los vectores columna $\mathbf{x_i} - \mu_{x_i}$ formando la matriz $\mathbf{X}$, la covarianza entre todas las columnas se llama *matriz de covarianza* y se expresar como:

$$\mathbf{C} = \frac{\mathbf{X}^t\mathbf{X}}{n-1} \tag{3}$$

Esta matriz es simétrica y semidefinida positiva por lo cual sus autovalores son no negativos.

Análisis de componentes principales

Consideremos una matriz de datos $\mathbf{X} \in \mathbb{R}^{m\times n}$ cuyas filas representan m vectores de dimensionalidad n. El análisis de componentes principales (PCA) busca encontrar un cambio de base de los vectores, de tal manera que las dimensiones se ordenen en componentes que explican los datos de mayor a menor. De forma análoga, el cambio de base busca ordenar las dimensiones según su varianza, de mayor a menor. Para hacer esto se realiza una descomposición en autovectores y autovalores de la matriz de covarianza de los datos:

$$\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^t \tag{4}$$

Donde $\mathbf{V}$ es la matriz con los autovectores en las columnas y $\mathbf{D}$ una matriz diagonal con los autovalores. Estos autovalores son la varianza que captura cada nueva dirección dada por los autovectores. La matriz $\mathbf{V}$ permite transformar los datos $\mathbf{X}$ en la nueva base mediante la operación $\mathbf{XV}$.

Dependiendo de la situación es posible reducir la dimensionalidad de los datos utilizando solo las p componentes principales que se deseen, es decir utilizando los primeros p vectores columna de la matriz $\mathbf{V}$.

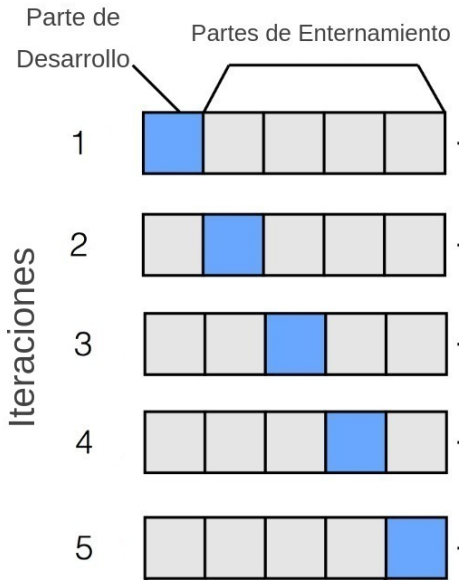

Figura 1: Validación Cruzada 5-fold

Para establecer cuantas p componentes son necesarias para obtener cierta aproximación, se suele considerar cuanta varianza acumulan las primeras p componentes, y poner un criterio, por ejemplo para mantener el 95 % de la varianza necesito 10 componentes.

La base formada por las componentes principales aprendidas sobre un conjunto de datos, pueden utilizarse para proyectar otro conjunto de datos. En ese sentido PCA es un modelo que aprende parámetros sobre un conjunto de datos de entrenamiento y puede aplicarse a datos nuevos no vistos. Es necesario realizar eso para garantizar obtener medidas de performance que indiquen generalización.

Validación cruzada

Los algoritmos que presentamos como KNN y PCA, ambos tienen hiperparámetros. En el primero es k la cantidad de vecinos y en el segundo p la cantidad de componentes aceptadas.

Hay que tomar ciertos recaudos a la hora de realizar una experimentación y explorar estos parámetros. Por ejemplo, si se utiliza un conjunto de datos de entrenamiento y otro de prueba, y se realiza una exploración que indica que cierta configuración de los hiperparámetros tiene la mejor performance en el conjunto de prueba, estaríamos obteniendo un resultado poco generalizable, sesgado y demasiado optimista. Es decir, no podemos saber si esa configuración de hiperparámetros será la mejor para conjuntos de datos no vistos.

Para soslayar esta situación, introducimos el concepto de validación cruzada[1] (*cross-validation* en ingles), este es un procedimiento que permite medir la performance de forma más generalizable.

En este trabajo utilizaremos la versión *k-fold cross-validation*. La idea es generar un conjunto de datos nuevo llamado de desarrollo (o confusamente llamado de validación) a partir de los datos de entrenamiento. Esto se puede hacer k veces a partir de una partición en

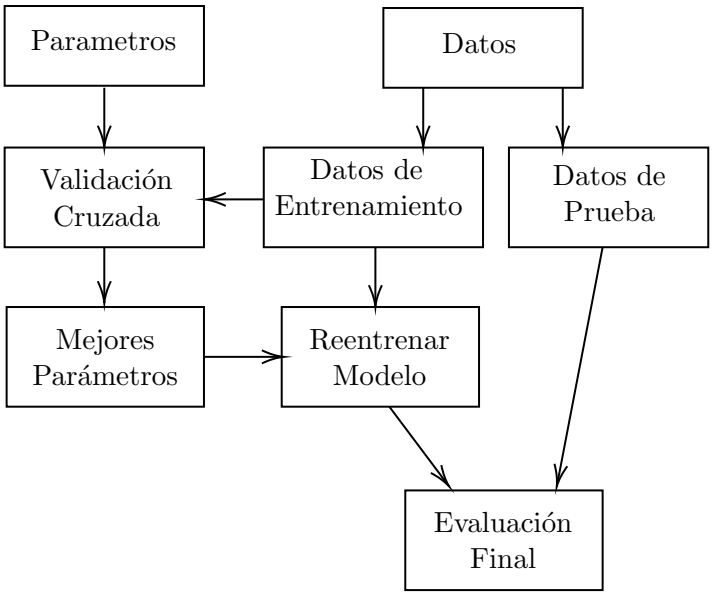

Figura 2: *Pipeline* de desarrollo, optimización de hiperparámetros y prueba

k partes del conjunto de entrenamiento. Supongamos 5 partes, entonces, se suele tomar 4 de ellas para entrenamiento y la restante para desarrollo. Esto se puede iterar de forma cíclica 5 veces, y de esta manera todos los datos se usaron como datos de desarrollo en algún caso (Fig. 1.

En el diagrama de la figura 2 podemos ver un *pipeline* en el cual se incluye el proceso de validación cruzada dentro de la optimización de hiperparámetros y de prueba.

Evaluación y optimización

Resumiendo, la tarea principal que vamos a querer resolver, es la de armar un clasificador de género de películas utilizando KNN y PCA. Para optimizar los parámetros k vecinos y p componentes, realizaremos validación cruzada. Reportaremos los mejores hiperparámetros encontrados y los resultados de la performance tanto para los conjuntos de desarrollo como de prueba.

Para medir la performance usaremos la siguiente medida.

$$\text{exa} = \text{exactitud} = \frac{\text{cantidad de muestra bien reconocidas}}{\text{cantidad de muestras totales}}$$

Podemos pensar una función única para especificar el problema completo que toma argumentos k vecinos, p componentes, y los conjuntos de entrenamiento $\mathcal{C}_{\text{entrena}}$ y de prueba $\mathcal{C}_{\text{prueba}}$:

$$\text{exa}\left(k, p, \mathcal{C}_{\text{entrena}}, \mathcal{C}_{\text{prueba}}\right)$$

Queremos encontrar los mejores k^* y p^* para el conjunto de desarrollo y obtener el resultado para el de prueba evaluando $\text{exa}\left(k^*, p^*, \mathcal{C}_{\text{entrena}}, \mathcal{C}_{\text{prueba}}\right)$

Los datos de desarrollo se generan para cada *fold*.

$$\mathcal{C}_{\text{des}}{}^{(i)}, \mathcal{C}_{\text{entrena}}{}^{(i)} = \text{xval}\left(\mathcal{C}_{\text{entrena}}, i, 5\right) \qquad 1 \leq i \leq 5$$

Para encontrar los mejores k y p optimizamos sobre el promedio de la exactitud de desarrollo.

$$\begin{aligned}
\text{exa}(k, p, \mathcal{C}_{\text{entrena}}, \mathcal{C}_{\text{des}}) &= \frac{1}{5} \sum_{i=5} \text{exa}\left(k, p, \mathcal{C}_{\text{entrena}}{}^{(i)}, \mathcal{C}_{\text{des}}{}^{(i)}\right) \\
k^*, p^* &= \arg\max_{k,p} \text{exa}(k, p, \mathcal{C}_{\text{entrena}}, \mathcal{C}_{\text{des}}) \\
\text{exa}_{\text{prueba}} &= \text{exa}\left(k^*, p^*, \mathcal{C}_{\text{entrena}}, \mathcal{C}_{\text{prueba}}\right)
\end{aligned}$$

Tareas

1. KNN

a) Implementar KNN usando *Python* y *Numpy*. Para poder aprovechar las operaciones vectorizadas de *Numpy* utilizaremos la distancia coseno para hacer la búsqueda de vecinos. Distancia coseno:

$$D(x,y) = 1 - Corr(x,y) = 1 - \frac{(\mathbf{x} - \mu_x) \cdot (\mathbf{y} - \mu_y)}{\sqrt{(\mathbf{x} - \mu_x) \cdot (\mathbf{x} - \mu_x)(\mathbf{y} - \mu_y) \cdot (\mathbf{y} - \mu_y)}}$$

Notar que es el producto escalar de los vectores con media cero puede hacerse de forma matricial entre los datos de entrenamiento y los de desarrollo o prueba.

Luego para encontrar los primeros k vecinos valerse de la función *np.argsort* y del indexado de arreglos que ofrece *Numpy*.

Luego la moda de las k etiquetas puede obtenerse con *scipy.stats.mode* y finalmente el computo de la performance total es un promedio sobre un vector que para cada dato de prueba dice si acertó o no.

2. Método de la potencia con deflación

a) Implementar en `C++` el método de la potencia con deflación para el cómputo de autovectores y autovalores. Utilizar como criterio de parada la norma infinito entre los vectores junto con un umbral de tolerancia. Como argumentos de entrada se puede optar tomar un archivo de texto con la matriz, la cantidad de iteraciones y la tolerancia para la convergencia, o bien utilizar una interfaz con Python. Como salida debe entregar los autovalores y autovectores como columnas de una matriz. Si desean pueden utilizar *Eigen* de `C++` —para trabajar con las matrices, y *ctypes* o *pbind11*[2] para interfacear con Python.

Por último, realizar tests para verificar la implementación del método en casos donde los autovalores y autovectores sean conocidos de antemano usando el truco de la matriz de Householder.

b) Estudiar la convergencia del método de la potencia. Para esto mediremos la cantidad de pasos que tarda en converger y una medida del error del método dada por $\|Av_i - \lambda_i v_i\|_2$ siendo v_i, λ_i la aproximación al autovector y autovalor i.

Para simplificar probaremos con una matriz dada con los siguientes autovalores $\{10, 10 - \epsilon, 5, 2, 1\}$ y haremos varias mediciones con matrices de Householder aleatorias.

Queremos visualizar el error del método descripto antes y la cantidad de pasos necesario para converger usando una tolerancia de 10^-7. Hacer gráficos del error y los pasos para cada autovalor en función del ϵ (espaciar logarítmicamente entre 10^{-4} y 10^0). Como se van a realizar medidas repetidas con datos aleatorios, considerar graficar los promedios y usar el desvío estándar como una barra de error.

3. Clasificación de géneros

a) Realizar un clasificador de género de películas usando KNN para un k fijo de 5, usando un 80 % de los datos de entrenamiento dados y medir la performance con la medida de exactitud en el 20 % restante. Recordar separar los datos de forma que quede la misma proporción de género en cada separación. Usar todas las 9581 dimensiones puede ser costoso, evaluar quedarse con las Q más frecuentes ($Q \in 500, 1000, 5000$).

[2] https://git.exactas.uba.ar/metodos/pybind-eigen-ejemplos

b) Para cada Q. Explorar el hiper-parámetro k, usando *4-fold cross-validation* con el conjunto de entrenamiento. Mantener en cada parte/fold el mismo balance del tipo de géneros. Medir la performance con la medida de exactitud en cada parte y tomar el promedio para encontrar el mejor k.

c) Preprocesar los datos de entrenamiento con PCA, usando la implementación del método de la potencia para encontrar los autovectores de la matriz de covarianza. Visualizar la cantidad de varianza explicada en función de la cantidad de componentes p.

d) Pipeline final: Exploración conjunta de los hiper-parámetros k de KNN, y p de PCA ($Q = 1000$).

Usar *4-fold cross-validation* con el conjunto de entrenamiento manteniendo en cada parte/fold el mismo balance de géneros en cada fold. Entrenar PCA con los datos de entrenamiento, y usar esa base para los datos de entrenamiento y desarrollo. Medir la performance en cada parte y tomar el promedio para encontrar el mejor k y p.

Una vez seleccionado el mejor k y p, entrenar con todos los datos de entrenamiento y medir la performance en los datos de prueba. ¡¡Cuidado!! calcular las matrices para PCA solo en los datos de entrenamiento y usar estas para preprocesar los de prueba.

En todos los casos es **obligatorio** fundamentar los experimentos planteados, proveer los archivos e información necesaria para replicarlos, presentar los resultados de forma conveniente y clara, y analizar los mismos con el nivel de detalle apropiado. En caso de ser necesario, es posible también generar instancias artificiales con el fin de ejemplificar y mostrar un comportamiento determinado.

Puntos Opcionales

1. Transformar la matriz de conteo de palabras con el proceso TF-IDF (del inglés Term frequency – Inverse document frequency), aplicar la búsqueda de hiper-parámetros a esta nueva matriz de datos y reportar las diferencias.

2. Comparar KNN con distancia euclídea vs distancia coseno, realizando la búsqueda de hiper-parámetros para cada distancia.

3. Variante de validación cruzada. En vez de promediar las performances de cada particionado, combinar todas las partes de desarrollo y computar una sola vez la métrica. Comparar contra el valor calculado tomando el promedio más menos el desvío estándar.

Forma y Fecha de entrega

- Se debe entregar un informe hecho en Latex de no más de 16 carillas. Este debe incluir una introducción al problema, la solución a las consignas mostrando pseudocódigos con una explicación de su funcionamiento. Figuras o tablas para visualizar resultados junto

con sus interpretaciones. Las figuras y tablas tienen que estar referenciadas en el texto y tener una descripción. Una sección de conclusiones con un resumen y de lo mostrado y los principales resultados. Para más detalles ver el archivo pautas [3] de escritura de informes en el campus.

- Se debe entregar los códigos que generan los resultados, figuras, etc. Si se requieren procedimientos especiales para ejecutar el código deben ser descriptos.
- La entrega es el **Jueves 31 de Octubre 23:59hs**. Para la entrega se facilitará un formulario online donde deberán adjuntar el informe y los archivos (incluir en el nombre de los archivos el nombre del grupo).
- La fecha de reentrega será el **5 de Diciembre hasta las 23.59 hs**

Referencias

[1] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[3] `https://campus.exactas.uba.ar/mod/resource/view.php?id=354262`