DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico #2 — Análisis de Sentimiento

10 de octubre de 2019 — Métodos Numéricos

Integrante	LU	Correo electrónico
Catriel Omar D'Elia	???	catriel.delia@gmail.com
Gian Franco Lancioni	234/15	gianflancioni@gmail.com
Joaquín Perez Centeno	699/16	joaquinpcenteno@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria — Pabellón I
Intendente Güiraldes 2160 — C1428EGA
Ciudad Autónoma de Buenos Aires — Argentina
Tel/Fax: +54 11 4576 3359
http://exactas.uba.ar

Índice

1. Abstracto

Distintos sitios de reseñas cinematográficas permiten a los usuarios escribir sus propias criticas y dar una puntuación a las películas. Al combinar texto con una clasificación numérica, se conforma un conjunto de datos interesante para llevar a cabo análisis supervisado de sentimiento sobre el lenguaje de los usuarios.

En este informe se evalúa el desempeño de clasificadores basados en kNN sobre *bag of words* de reseñas de películas para clasificar el sentimiento subyacente en el texto. Adicionalmente se evaluó el desempeño de utilizar PCA en el pre procesamiento del texto.

FIXME acá hay que poner un adelanto de las conclusiones.

2. Introducción Teórica

2.1. Análisis de Sentimiento

2.2. Bag of Words

Bag of Words es un sistema de representación utilizado en procesamiento del lenguaje natural. Cada documento pasa a ser representado por su multiconjunto de palabras. Se pierde el orden original de las palabras en el texto. Se utiliza en problemas donde el vocabulario del documento es suficiente para estimar el sentimiento subyacente.

2.3. k Nearest Neighbors

El algoritmo *k Nearest Neighbors*, kNN es un clasificador supervisado que estima la categoría de una instancia basandose en su vecindad con instancias ya conocidas. Se desempeña mejor en problemas donde la vecindad espacial entre observaciones es un factor importante en la clasificación de instancias.

Para una instancia de entrada x, se calcula su distancia con todo el set de entrenamiento. La clasificación estimada resultante es la categoría modal (voto mayoritario) entre los k vecinos mas cercanos a x en el set de entrenamiento.

Influye en kNN la elección del k en el rango $1 \leq k \leq N$, así como la elección de la norma a utilizar para medir distancia. Para un valor de k alto, la estimación puede verse afectada por la categoría modal en el conjunto de entrenamiento. Para una eleción de k chico, la clasificación puede verse afectada por la presencia de outliers *outliers* en su cercanía. FIXME hablar de distintas normas.

2.4. Principal Component Analysis

Principal Component Analysis, PCA, es un algoritmo de reducción de dimensionalidad que transforma los datos de entrada en un espacio de *features* correlacionadas a un conjunto de valores cuyas *features* no guarden correlación entre sí.

Si del conjunto de vectores $\{v_1, ..v_n\} \subset \mathrm{R}^m$ que conseguimos mediante Bag of Words tomamos el vector de medias muestrales $\mu = \sum \frac{v_i}{n} \in \mathrm{R}^n$ podemos conseguir la matriz $X \in \mathrm{R}^{m \times n}$ con filas $\frac{v_i - \mu}{\sqrt{n-1}}$ podemos construir su *matriz de covarianza* $M = \frac{X^t X}{n-1}$ de cuya descomposición SVD, como para toda matriz X vale que $X^t X$ es simétrica con autovalores reales y base ortonormal de autovectores también reales, conseguimos una base de autovectores ordenada decrecientemente de acuerdo a la varianza interna de cada componente (que se asocia a qué tan grande es su autovalor) y con nula covarianza entre sí de modo que un cambio de base con los primeros autovectores pase la matriz a una base con menos ruido entre componentes.

Esto también nos permite recortar los últimos vectores de la base dado que son los que menos información proveen (además de ahorrar tiempo y espacio al considerar menos componentes espaciales en kNN), que llamermos una *reducción de dimensionalidad* a α cantidad de vectores.

2.5. Método de la potencia

Se trata de un método iterativo que permite hallar los autovalores dominantes de una matriz dada (aquellos cuya norma supera al resto) y sus autovectores asociados. O sea conseguimos un λ_1 tal que:

$$|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|$$

Para todos los autovalores λ_i en una matriz de $n \times n$.

La idea es usar este método para conseguir los autovectores de las α componentes más significativas descriptas en 2.4. Al tratarse de un método iterativo que converge asintóticamente tendremos que implementar distintos criterios de parada para dejar de iterar.

Usamos la variante del método de la potencia que se presenta para matrices simétricas (como ya vimos, la matriz de covarianzas cumple tal propiedad) vista en la sección 9.3 del libro de Burden [1].

3. Experimentación

3.1. Metodología

3.2. Convergencia de método de la potencia

Como en cada k-ésima iteración del método de la potencia estamos consiguiendo un $v^{(k)}$ autovector tentativo asociado a un $\hat{\lambda}^{(k)}$ autovalor también tentativo podemos analizar su evolución para determinar si están convergiendo y cortar la iteración cuando ya lo consideremos suficiente. Dado un $\epsilon \in \mathrm{R}$ tal que $\epsilon > 0$, la matriz cuadrada $A \in \mathrm{R}^{n \times n}$ sobre la cual buscamos su autovalores, y los autovalores y autovectores tentativos de la k-ésima iteración $\hat{\lambda}^{(k)} \in \mathrm{R}$ y $v^{(k)} \in \mathrm{R}^n$ planteamos los siguientes criterios de corte:

- Criterio de *vector residual*: cortar si $||Av^{(k)} - v^{(k)}\hat{\lambda}^{(k)}|| < \epsilon$
- Criterio de diferencia de autovectores: cortar si $||v^{(k)} - v^{(k-1)}|| < \epsilon$
- Criterio de diferencia de autovalores: cortar si $||\hat{\lambda}^{(k)} - \hat{\lambda}^{(k-1)}|| < \epsilon$

Como bajo ciertas condiciones (como $|\lambda_1| \approx |\lambda_i|$ para λ_1 autovalor dominante y λ_i otro autovalor de la matriz A) el método de la potencia puede no converger, al margen de los criterios se corta la iteración tras una cantidad fija y lo suficientemente grande (de modo que si converge no interrumpa a los criterios anteriores) de veces.

En cuanto a precisión especulamos originalmente con que el mejor fuera el criterio de vector residual dado que mide qué tan bien aproximan $\hat{\lambda}^{(k)}$ y $v^{(k)}$ a un autovalor y autovector asociado, seguido del criterio de autovectores bajo la presunción de que funciona mejor al tener que aproximar varias componentes del autovector para terminar versus aproximar únicamente el autovalor. Bajo las mismas suposiciones especulamos livianamente que la precisión sería inversamente proporcional a la cantidad de iteraciones necesarias siendo el criterio de diferencia de autovalores el que antes se cumpliría, seguido del de autovectores y por último el de vector residual.

Sin embargo, investigando más profundamente nos encontramos con el resultado del Teorema 9.19 del libro de Burden [1] que indica que si vale:

$$||Av^{(k)} - \hat{\lambda}^{(k)}v^{(k)}|| < \epsilon \tag{1}$$

entonces tenemos que, con λ_j autovalores de A:

$$\min_j |\lambda_j - \hat{\lambda}^{(k)}| < \epsilon \tag{2}$$

Si bien da una buena idea de que el criterio de vector residual contiene en sí información sobre la convergencia de los autovalores (tentando la idea de que es más restrictivo que el criterio de autovalores) no necesariamente el λ_j más cercano a $\hat{\lambda}^{(k)}$ sea el λ_1 buscado y también podría suceder que, bajo ciertas condiciones, $v^{(k)}$ esté más cerca de λ_1 que de $v^{(k-1)}$.

El criterio de diferencia de autovectores nos resultaba interesante, dado que no considera autovalores para cortar, de ser lo suficientemente bueno significaría solo computarlos al final del método y no en cada iteración, además de que se condice en espiritu con el uso que le vamos a dar al resultado (solamente usar los autovectores para cambiar de base la matriz de covarianzas, sin importar los autovalores)

Otro resultado interesante en la misma sección del mismo libro es que, como indican al hablar de aceleración de convergencia, la secuencia $\{\hat{\lambda}^{(k)}\}$ converge, cuando $\lambda_1 > \lambda_2$ con λ_2 el más grande (potencialmente no único) de los autovalores no dominantes, linealmente (más específicamente en tasa de convergencia $\mathcal{O}(\lambda_2/\lambda_1)$). Inicialmente entendimos mal que esto significaría que la cantidad de iteraciones fuera lineal sobre ϵ sino que, por el contrario, que la tasa de convergencia sea lineal significa justamente que la sucesión se

aproxima de manera lineal a λ_1 (ver definición en sección del libro [1]), lo que implica una convergencia lenta con potencialmente muchas más iteraciones que una secuencia de convergencia, por ejemplo, cuadrática. De hecho, como se puede ver en la tabla 2.7 [1], una secuencia de tales características es potencialmente exponencial incluso.

Para experimentar usamos método de la potencia con los 3 criterios de manera separada moviendo con 20 iteraciones (sobre las que se considera promedio y desvio estandar) por valor de ϵ en un rango de 1 a 10^{-14} (iteramos el exponente i tal que $10^{-i} = \epsilon$ entre 0 y 14), el primer número resultado de una corrida en la que verificamos que el error era muy grosero como para aproximar bien autovectores y el último número resultando de probar a mano y ver que el tiempo que tardaba no lo considerábamos admisible para una ejecución cómdo (teniendo en cuenta que luego en vecinos más cercanos y PCA ibamos a iterar continuas veces el método), buscando un punto intermedio de precisión y latencia aceptables. Para medir el error usamos la inversa (de modo que sea creciente conforme mejor precisión) de la norma del mismo vector residual $||Av^{(k)} - v^{(k)}\hat{\lambda}^{(k)}||$ dado que por su naturaleza de medir 'qué tan buenos autovectores y autovalores' son nos resultó más atractiva que la idea de comparar contra librerias como *NumPy*.

Inicialmente probamos con un set reducido de 2000 vectores (elegidos aleatoriamente sobre el total) provenientes del dataset *imdb_small.csv* sobre los que se arma la matriz de covarianzas para poder determinar si podíamos recortar un poco más el intervalo antes de pasar al dataset entero. Consiguiendo los siguientes resultados:

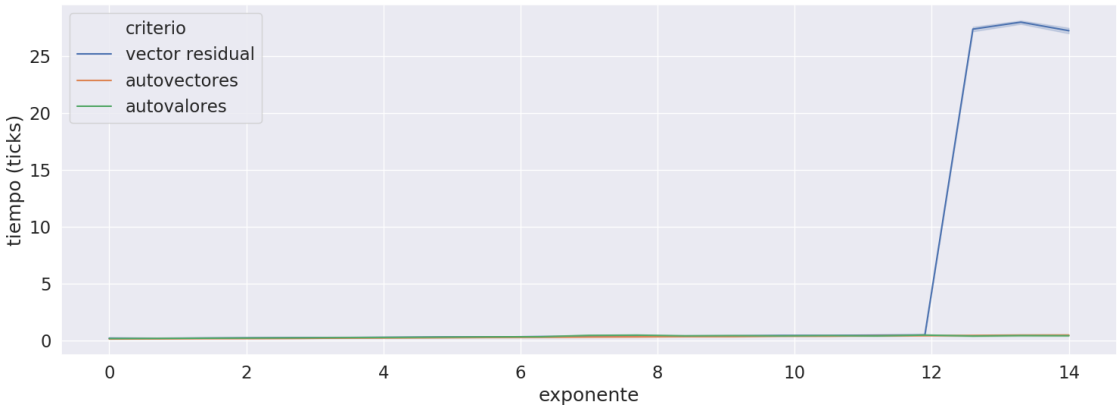

Figura 1: Progresión del tiempo medido en ticks del metodo de la potencia en función del exponente del epsilon para 2000 vectores aleatorios.

Como se puede ver en la figura 1 la latencia del criterio vector residual explota desmedidamente a partir del exponente 12, como el tiempo era demasiado decidimos recortar el rango y mover el exponente solo hasta 12 antes de pasar a medir con el dataset completo.

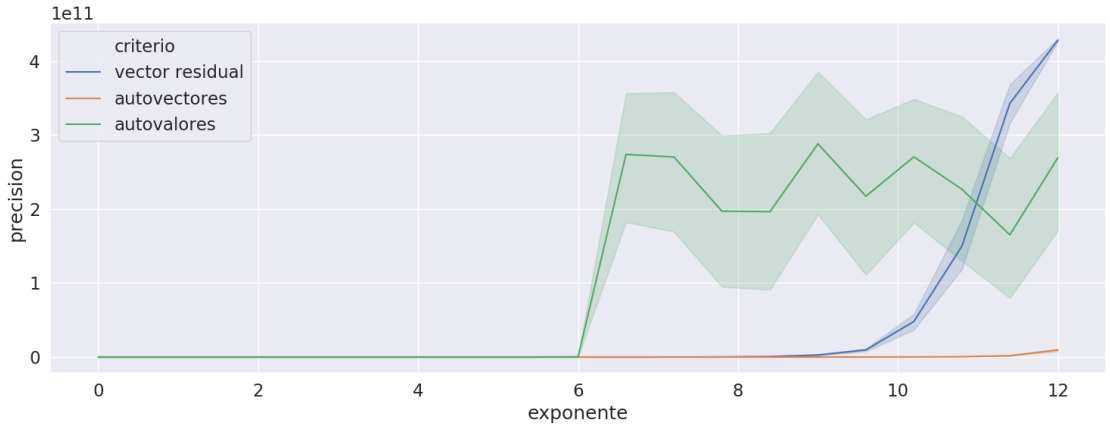

Figura 2: Progresión de precisión del metodo de la potencia en función del exponente del epsilon para los 6225 vectores del dataset.

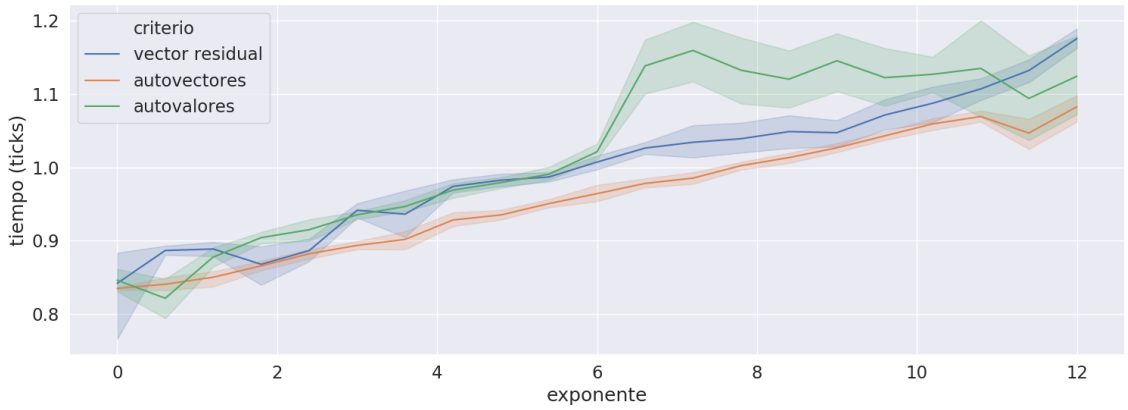

Figura 3: Progresión del tiempo medido en ticks del metodo de la potencia en función del exponente del epsilon para 6225 vectores del dataset.

Vemos en la figura 2 que el criterio de autovalores alcanza precisiones considerables con menos exponentes que el resto pero se mantiene inestable en variaciones y con media constante antes de que el criterio de vector residual empiece a subir de manera mucho más consistente. El criterio de autovalores recien sobre los últimos exponentes empieza a mostrar una mínima mejora.

El problema de la explosión rápida en precisión del criterio de autovalores es que viene acompañado de una explosión en latencia como indica la figura 3 y en el caso del criterio de vector residual para el momento en que alcanza una precisión similar al criterio de autovalores la latencia ya es considerable (tener en cuenta que la linealidad en función del exponente significa exponencialidad respecto de epsilon para la latencia). Decidimos entonces, antes de analizar los exponentes mayores a 6 tras la explosión del criterio de autovalores y en la subida del criterio de autovectores, analizar qué sucede antes del 6.

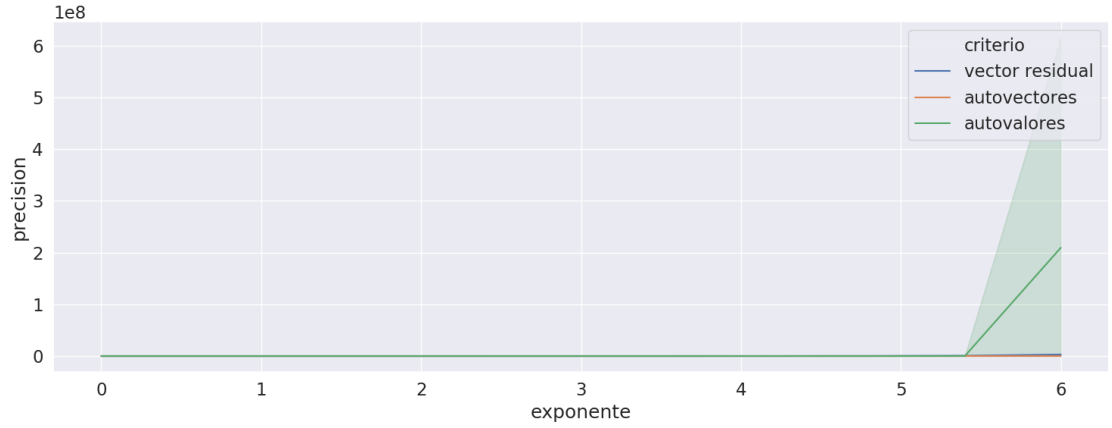

Figura 4: Progresión de precisión del metodo de la potencia en función del exponente del epsilon para los 6225 vectores del dataset.

Viendo las figuras 5 y 4 consideramos que el exponente 6 era una interesante alternativa dado que tiene en autovalores latencia mucho menor a los exponentes mayores y aun así mostraba una mejora importante de precisión respecto de exponentes menores. Decidimos entonces, antes de pasar a comparar los valores mayores a 6 la posibilidad de comparar el criterio de autovalores para exponente 6 contra estos (particularmente 7, primer valor tras la explosión) en una instancia de PCA con k y α fijos arbitrarios, dado que solamente nos interesa medir qué tan bien funcionan en un caso práctico de PCA (que es la finalidad del método) y ninguno de los dos parámetros favorece a ninguno de los dos exponentes (sí afecta α en cómo se arrastra errores de precisión en el método por cuestiones de deflación, pero penaliza justamente al menos preciso de modo que no nos afecta). Al iterar corridas con tales parámetros y notar que la medida de accuracy era exactamente la misma en ambos casos con tiempos favorables (cerca de la mitad en algunos casos) para el menor exponente decidimos entonces fijar $\epsilon = 10^{-6}$.

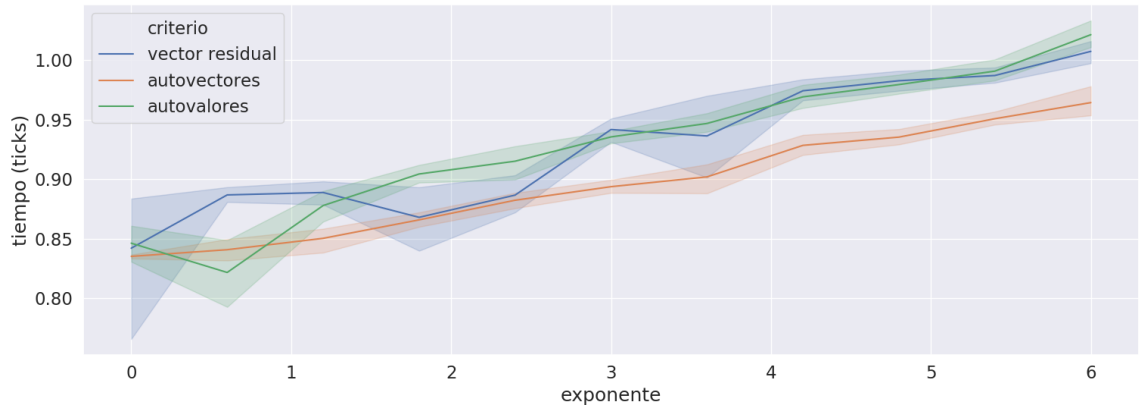

Figura 5: Progresión del tiempo medido en ticks del metodo de la potencia en función del exponente del epsilon para 6225 vectores del dataset.

3.3. Optimización de k para kNN sin PCA

Para buscar una k cantidad de vecinos que optimice kNN usamos nuevamente el dataset de *imdb_small*, iteramos un k de 1 a 3000 con saltos de a 20.

Originalmente pensábamos arrancar en un rango mas adelantado dado que para k pequeños sobre muestras grandes se puede correr riesgos de overfitting por puntos ruidosos"que estén demasiado pegados ganandole en la votación a la verdadera clase, pero como k pequeños no son caros de computar los consideramos en la experimentación de todos modos.

Elegir k grande tiene el problema de que los elementos de la clase de un punto quizás están todos en su vecindad inmediata y a medida que vamos buscando vecinos en zonas más alejadas vamos tendiendo a clasificar peor el punto: una clase muy densa pero rodeada de otra mas dispersa y sobrerrepresentada será peor catalogada cuanto más grande sea k. Si además k es lo suficientemente grande (aproximadamente superando la mitad de la población total) la proporción de vecinos más cercanos se parece cada vez más y más a la proporción muestral de cada clase sobre el total, lo cual no aporta nada de información. Por lo tanto decidimos iterar solamente hasta la mitad.

Tendría sentido con este análisis que acabamos de hacer encontrar los mejores resultados en un punto intermedio en la magnitud de k.

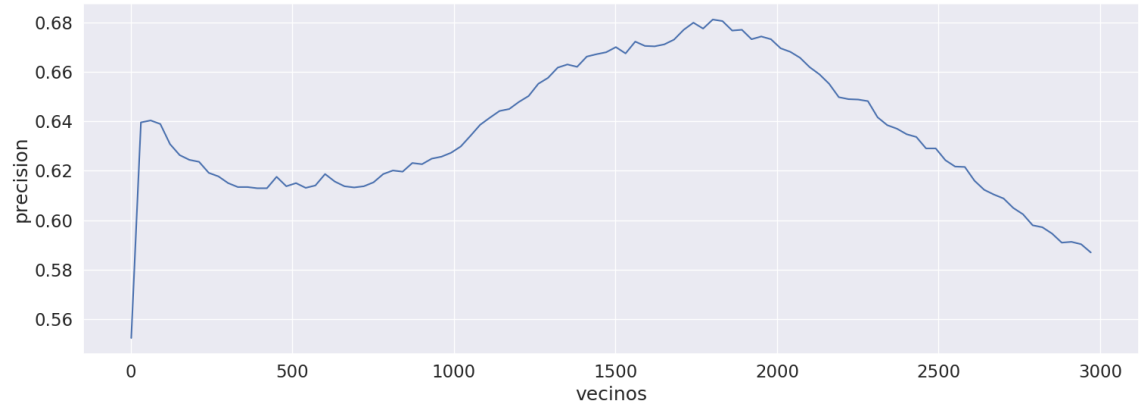

Figura 6: Progresión de la precisión de KNN en función de la cantidad de vecinos.

Como podemos apreciar en la figura 6, se cumple nuestra predicción de que la precisión empeora conforme la cantidad de vecinos se vuelve muy grande o muy pequeña. Alcanzamos el máximo accuracy score en 0.681116 para $k = 1801$ que se sitúa muy cerca de la mitad del intervalo que elegimos. Nos resulta complicado comprender por qué desciende la precisión en los primeros valores, especulamos con que los problemas de ruido y sesgo requieran valores de k bajos pero mayores a los mínimos para que empiecen a aparecer. Otro motivo sea que si el espacio de los vectores no esté separado de manera clara existan zonas con más ruido

que en el resto y que se vayan acumulando outliers de clases difusas hasta que se sale de los mismos.

3.4. Optimización de k y α para kNN con PCA

4. Conclusiones

5. Referencias

Referencias

[1] Burden, Richard L. and J. Douglas Faires. *Numerical analysis. Fifth Edition.* Boston: PWS Publishing Company, 1993.