Métodos Numéricos
1er Cuatrimestre 2019
Trabajo Práctico 2

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

Análisis de Sentimiento

Introducción

Poder extraer información de lo que dicen los usuarios en Internet es un tema que está muy en boga desde hace ya unos cuantos años y ha tenido su boom con la eclosión de las Redes Sociales [5]. Un uso potencial de estas opiniones se encuentra en lo comercial: ¿Qué opinan los televidentes de la nueva serie de Adrián Suar? ¿Hablan en Twitter positivamente de la nueva Coca-Cola? ¿Gustó la nueva película de Nicholas Cage?

Los intereses sobre este campo trascienden lo comercial, y la posibilidad de analizar cuantitativamente las opiniones de los usuarios (y votantes) despierta también intereses políticos: ¿Sobre qué candidato tienen mejor opinión los votantes? ¿A qué referente político atacan más en las redes sociales? Estas son preguntas que atacan muchas consultoras. Hace ya más de 10 años que se estudia la incidencia de las campañas en Internet y de las opiniones de los usuarios [3, 6]. Recientemente el escándalo de *Cambridge Analytica* en las elecciones presidenciales del 2016 en EEUU ha dado cuenta del uso de estas tecnologías con fines electorales.

El Análisis de Sentimiento (también conocido como *Opinion Mining*) se refiere al conjunto de técnicas de Procesamiento de Lenguaje Natural (NLP), Lingüística Computacional e Inteligencia Artificial utilizadas para poder extraer, identificar y cuantificar información acerca del estado de ánimo de un sujeto. En particular, nos interesa analizar la polaridad de un texto; es decir, si un texto tiene una emoción "positiva" o "negativa" en líneas generales.

En este trabajo práctico construiremos un analizador de polaridad para las opiniones sobre películas de usuarios de IMDB[1]. Para ello, construiremos un clasificador basado en la técnica de vecinos más cercanos (*K-Nearest Neighbors*) y reducción de la dimensionalidad con análisis de componentes principales (*Principal Component Analysis*).

Metodología

Se requiere desarrollar un algoritmo de detección de polaridad para un conjunto de datos (*dataset*) de reseñas de IMDB; esto es: dado el texto de una reseña de una película, queremos poder decir si esta es negativa o positiva.

Para ello, contaremos con un dataset de reseñas etiquetadas en *positivas* y *negativas*. El algoritmo se *entrenará* sobre un subconjunto de esas reseñas (*conjunto de entrenamiento*). Luego, para verificar su efectividad, corroboraremos con otro subconjunto disjunto (*conjunto de testing*) los resultados que nos dé sobre cada una de las reseñas.

Como algoritmo de clasificación, utilizaremos *k vecinos más cercanos* (kNN, k-Nearest Neighbors) [1]. En su versión más simple, este algoritmo considera a cada instancia de entrenamiento (en nuestro caso, cada reseña) como un punto en el espacio euclídeo m-dimensional. Cuando querramos clasificar una instancia como positiva o negativa, buscaremos las k instancias de entrenamiento más cercanas, y le daremos la etiqueta mayoritaria entre esos vecinos.

[1] Internet Movie DataBase, `https://www.imdb.com/`

Modelo de Bolsa de palabras

La entrada de *kNN* deben ser vectores de $\mathbb{R}^m$, para un m fijo, uno por cada instancia. Para convertir cada reseña en un vector de longitud fija utilizamos el modelo de Bolsa de Palabras (*Bag of Words* o BoW). El mismo requiere contar con un vocabulario ordenado $\mathcal{V}$ de m palabras (de forma no necesariamente lexicográfica).

Luego, para una instancia dada x, se analiza el texto correspondiente contando cuántas veces apareció cada palabra del mencionado vocabulario. El texto x se representará como un vector $x_b = BoW(x)$, $x_b \in \mathbb{R}^m$ donde en la coordenada i-ésima, x_b tendrá cuántas veces apareció la palabra número i del vocabulario en el texto.

Por ejemplo, si mi vocabulario ordenado es:

$$\mathcal{V} = [chicos, chicas, y, quieren, rock, falta, te] \tag{1}$$

El texto "chicos y chicas quieren rock, quieren rock" lo codificaríamos como el vector $[1, 1, 1, 2, 2, 0, 0]$. El texto "te falta rock" lo codificaríamos como el vector $[0, 0, 0, 0, 1, 1, 1]$.

El problema que tiene este método es que si tenemos un vocabulario demasiado grande[2], genera vectores de igual dimensión. Para sortear este problema, se suelen filtrar dos tipos de palabras:

- Aquellas con frecuencia demasiado alta: casi siempre preposiciones, conjugaciones de verbos comunes.
- Palabras con muy baja frecuencia

Esas palabras no aportan información significativa para la mayoría de estos problemas, y no nos permiten encontrar un patrón común entre las diferentes clases a separar (en nuestro caso, reseñas positivas y negativas). Parte de la experimentación consistirá en poner umbrales para filtrar las palabras y ver cuál da mejores resultados.

Procedimiento de k vecinos más cercanos

El algoritmo de clasificación con *kNN* se puede resumir en los siguientes pasos:

- Se define un dataset de entrenamiento como el conjunto $\mathcal{D} = \{x_i \ : \ i = 1, \ldots, n\}$, donde cada x_i tiene asociado un número entre 1 y la cantidad de clases.
- Luego, se define m como el número total de dimensiones de cada instancia x_i representada como un vector $x_i \in \mathbb{R}^m$.
- De esta forma, dado un nuevo $x \in \mathbb{R}^m$, tal que $x \notin \mathcal{D}$, para clasificarlo 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.

[2]Un vocabulario estándar puede contar con más de 160.000 palabras.

El algoritmo de los vecinos más cercanos es muy sensible a la dimensión de los objetos y a la variación de las componentes del vector instancia. Es por eso, que las instancias dentro del conjunto de datos $\mathcal{D}$ se suelen *preprocesar* para lidiar con estos problemas. Estas instancias se pueden considerar como vectores que se encuentran en un espacio de dimensión alta, lo cual suele traer dificultades para realizar cálculos y para diseñar algoritmos de reconocimiento que puedan utilizar la información que representan.

Teniendo en cuenta esto, una alternativa interesante de preprocesamiento es buscar reducir la cantidad de dimensiones de las muestras para trabajar con una cantidad de variables más acotada buscando que las nuevas variables tengan información representativa para clasificar los objetos de la base de entrada.

En esta dirección, consideraremos el método de reducción de dimensionalidad *análisis de componentes principales* o PCA (por sus siglas en inglés) dejando de lado los procesamientos de datos que se puedan realizar previamente o alternativamente a aplicar PCA.

Análisis de componentes principales

El método de análisis de componentes principales o PCA consiste en lo siguiente.

Sea $\mu = (x_1 + \ldots + x_n)/n$ el promedio coordenada a coordenada de los datos $\mathcal{D} = \{x_i : i = 1, \ldots, n\}$ tal que $x_i \in \mathbb{R}^m$. Definimos $X \in \mathbb{R}^{n \times m}$ como la matriz que contiene en la i-ésima fila al vector $(x_i - \mu)^t/\sqrt{n-1}$. La matriz de covarianza de la muestra X se define como $M = X^t X$.

Siendo v_j el autovector de M asociado al j-ésimo autovalor, al ser ordenados por su valor absoluto, definimos para $i = 1, \ldots, n$ la *transformación característica* de x_i como el vector $\mathbf{tc}(x_i) = (v_1\, x_i, v_2\, x_i, \ldots, v_\alpha\, x_i)^t \in \mathbb{R}^\alpha$, donde $\alpha \in \{1, \ldots, m\}$ es un parámetro de la implementación. Este proceso corresponde a extraer las α primeras *componentes principales* de cada muestra. La idea es que $\mathbf{tc}(x_i)$ resuma la información más relevante de la muestra, descartando los dimensiones menos significativas.

El método PCA previamente presentado sirve para realizar una transformación de los datos de entrada a otra base y así trabajar en otro espacio con mejores propiedades que el original.

Procedimiento para la clasificación con kNN y PCA

Una vez elegidos los parámetros k del kNN y α de PCA, el proceso completo de clasificación de reseñas se puede resumir como:

1. Ingresar una nueva reseña x no presente en el conjunto de entrenamiento $\mathcal{D}$.
2. Computar el $x_b = BoW(x)$ utilizando el vocabulario $\mathcal{V}$.
3. Calcular $\mathbf{tc}(x_b)$ la transformación característica.
4. Comparar con cada $\mathbf{tc}(x_{bi})$, $\forall x_{bi} \in \mathcal{D}_b$ donde $\mathcal{D}_b$ es el dataset de training preprocesado con BoW.
5. Elegir la moda entre los k vecinos más cercanos.

6. Devolver la clase de la moda ("pos" o "neg") como el resultado de la clasificación.

Enunciado

Se pide implementar un programa que lea desde archivos las instancias de entrenamiento correspondientes y que, utilizando los métodos descriptos en la sección anterior, para una nueva instancia determine a qué clase pertenece.

Para ello, el programa **deberá** implementar el algoritmo de kNN así como también la reducción de dimensión utilizando PCA.

Con el objetivo de obtener las transformaciones características de cada método, **se deberá** implementar el método de la potencia con deflación para la estimación de autovalores/autovectores de la matriz de covarianza en el caso de PCA. Además, se **deberá** analizar el criterio de convergencia en función de la precisión obtenida y tiempo de cómputo.

El programa principal puede estar hecho en python pero **es requisito** implementar en `C++` el algoritmo de kNN, PCA y el método de la potencia.

Se recomienda realizar tests para verificar la implementación del método de la potencia en casos donde los autovalores y autovectores sean conocidos de antemano. También, puede resultar de utilidad emplear Python, Matlab/Octave o alguna biblioteca de cálculo numérico para verificar los resultados.

Es **requisito** trabajar con el modelo de bolsa de palabras descripto anteriormente y estudiar los mejores umbrales para filtrar las palabras que reporten mejores resultados.

En todos los casos, se **deberá** trabajar al menos la base de datos provista por la cátedra descripta en el documento adjunto.

Experimentación

Para guiar la experimentación, se **pide** realizar lo siguiente:

- Analizar la calidad de los resultados obtenidos al combinar kNN con y sin PCA, para un rango amplio de combinaciones de valores de k y α. Llamamos k a la cantidad de vecinos a considerar en el algoritmo kNN y α a la cantidad de componentes principales a tomar.
- Analizar la calidad de los resultados obtenidos al combinar kNN con PCA, para un rango amplio de instancias de entrenamiento. Utilizar desde muy pocas hasta todas las disponibles para identificar en qué situación se comporta mejor cada uno de los métodos.
- ¿Cómo se relaciona k con el tamaño del conjunto de entrenamiento? Pensar el valor máximo y mínimo que puede tomar k y qué sentido tendrían los valores.
- En base a los resultados obtenidos para ambos métodos, seleccionar aquella combinación de parámetros que se considere la mejor alternativa, con su correspondiente justificación, compararlas entre sí y sugerir un método para su utilización en la práctica.

La calidad de los resultados de clasificación obtenidos será analizada mediante diferentes métricas:

1. Accuracy
2. Precision/recall
3. F1-Score

En particular, la métrica más importante que **debe** reportarse en los experimentos es la tasa de efectividad lograda o *accuracy*.

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 (no obligatorios)

- Proponer y/o implementar alguna mejora al algoritmo de kNN. En particular pensar en distintas distancias posibles para los vecinos.
- Aplicar n-gramas (ver [2] para una explicación).
- Realizar un estudio experimental de los métodos propuestos sobre una base de entrenamiento utilizando la técnica *K-fold cross validation* mencionada anteriormente, con el objetivo de analizar el poder de clasificación y encontrar los mejores parámetros de los métodos.
    - Justificar el por qué de la elección de los mismos. Tener en cuenta que probar todos los valores posibles puede ser muy costoso.
    - ¿En qué situaciones es más conveniente utilizar K-fold con respecto a no utilizarlo?
    - ¿Cómo afecta el tamaño del conjunto de entrenamiento?

Formato de entrada/salida

El programa ejecutable a entregadar **deberá** contar con las funcionalidades pedidas en este apartado. El mismo deberá tomar al menos tres parámetros por línea de comando con la siguiente convención:

```
$ ./tp2 -m <method> -d <dataset_path> -o <classif>
```

donde:

- `<method>` es el método a ejecutar con posibilidad de extensión (0: kNN, 1: PCA + kNN, ... etc)

- `<dataset_path>` es el nombre del archivo de entrada con los datos de entrenamiento y los datos de testeo a clasificar. Este debe contener múltiples líneas con el siguiente formato:

  `<id>,<dataset>,<etiqueta>,<tokens>`

  donde

  - `<id>` es el id de la reseña
  - `<dataset>` es `test` o `train` según corresponda
  - `<etiqueta>` es `neg` o `pos` según corresponda
  - `<tokens>` es una lista de tokens separados por coma

  Utlizar el archivo `imdb_tokenized.csv` a modo de ejemplo.

- `<classif>` el nombre del archivo de salida. Este archivo deberá tener tantas líneas como entradas de testing hayan en el dataset que se utilice. Cada una deberá tener el id de la reseña que haya sido clasificada, así como la clase (positiva o negativa) a la cual fue asignada.

Un ejemplo de invocación sería el siguiente:

```
$ ./tp2 -m 1 -d datos/imdb_tokenized.csv -o result.csv
```

Además, el programa deberá imprimir por consola un archivo, cuyo formato queda a criterio del grupo, indicando la tasa de reconocimiento obtenida para cada conjunto de test y los parámetros utilizados para los métodos.

Nota: cada grupo tendrá la libertad de ampliar las funcionalidades provistas por su ejecutable manteniendo el formato de obligatorio mencionado. En particular, puede ser de utilidad alguna variante de toma de parámetros que permita entrenar con un porcentaje de la base de datos de entrenamiento y testear con el resto (ver archivos provistos por la cátedra). Además, puede ser conveniente separar la fase de entrenamiento de la de testeo/consulta para agilizar los cálculos.

Fecha de entrega

- *Formato Electrónico:* domingo 19 de mayo hasta las 23.59 hs, enviando el trabajo (informe + código) a la dirección `metnum.lab@gmail.com`.

  - El subject del email debe comenzar con el texto `[TP2]` seguido de la lista de apellidos de les integrantes del grupo separados por punto y coma `;`.
    Ejemplo: `[TP1] Lennon; McCartney; Starr; Harrison`

  - Se ruega no sobrepasar el máximo permitido de archivos adjuntos de 20MB. Tener en cuenta al realizar la entrega de no ajuntar bases de datos disponibles en la web, resultados duplicados o archivos de backup.

- *Recuperatorio:* jueves 6 de junio hasta las 23.59 hs, enviando el trabajo corregido a la dirección `metnum.lab@gmail.com`
- Pautas de laboratorio: `https://campus.exactas.uba.ar/pluginfile.php/126018/course/section/17889/pautas.pdf`

Importante: El horario es estricto. Los correos recibidos después de la hora indicada serán considerados re-entrega.

Referencias

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

[2] Daniel Jurafsky. Speech and language processing: An introduction to natural language processing. *Computational linguistics, and speech recognition*, 2000.

[3] Matthew James Kushin and Masahiro Yamamoto. Did social media really matter? college studentsúse of online media and political decision making in the 2008 election. *Mass Communication and Society*, 13(5):608–630, 2010.

[4] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[5] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[6] Lee Rainie and John Horrigan. Election 2006 online. *Pew Internet & American Life Project Report*, 2007.