Fundamentos de Programación

Práctica 1 - Curso 2018/2019

Plazo de entrega: 29 de marzo de 2019

Clasificación de e-mails mediante distribuciones de probabilidad

En primer lugar, se escribirá una función en lenguaje Python que calcule el histograma de palabras de un e-mail:

- La función tomará como entrada el nombre de un archivo de texto, que contendrá un e-mail sobre un cierto tema o clase (deportes, religión, historia, etc.), estructurado en dos partes: cabecera y contenido. La cabecera consistirá en una serie de líneas de la forma:

  <item-id>: <item-value>

  seguidas de una línea vacía. A continuación comenzará propiamente el contenido del e-mail.

- La función deberá eliminar todos los elementos de la cabecera, y procesar el contenido del e-mail como sigue:
  - Se eliminarán todos los símbolos de puntuación.
  - Se eliminarán todas las palabras que contengan símbolos no alfabéticos (números, caracteres *extraños* como '@', '#', etc.).
  - Todas las palabras se pasarán a minúsculas.
- Las palabras observadas en un e-mail se utilizarán para construir un diccionario de cuentas (un histograma), que constituirá el valor de retorno de la función.

En segundo lugar, se escribirán en lenguaje Python dos funciones auxiliares:

- Una función que acumule las cuentas de un histograma en otro histograma. Esta función debería servir para ir acumulando las cuentas de los e-mails de una clase, de modo que al final se obtenga el histograma de la clase.
- Una función que tome como entrada un histograma y normalice sus cuentas para producir lo que denominamos *frecuencias*. Es decir, si $h(w)$ es la cuenta de una palabra w, la cuenta normalizada o frecuencia, $h_{norm}(w)$, se calculará como sigue:

$$h_{norm}(w) = \frac{h(w)}{\sum_{\forall v \in h} h(v)} \tag{1}$$

  El diccionario resultante h_{norm} representa una distribución de probabilidad definida sobre el conjunto de palabras de la lengua. Si el histograma de partida ha sido calculado a partir de un gran número de e-mails de una clase c, entonces dicha distribución puede usarse como modelo: $h_{norm}(w)$ representaría la probabilidad de observar una cierta palabra w en e-mails de la clase c. Según este modelo, una palabra que no aparece en los e-mails con los que se ha calculado el histograma de la clase c, tendrá probabilidad nula de aparecer en dicha clase. En todo caso, se verifica que:

$$\sum_{\forall w \in h_{norm}} h_{norm}(w) = 1 \tag{2}$$

El objetivo de esta práctica es medir la precisión que se obtiene al clasificar e-mails mediante distribuciones de probabilidad definidas sobre el conjunto de palabras de la lengua. Para ello, deberá escribirse un programa en lenguaje Python que estime las distribuciones de probabilidad de las distintas clases, utilizando un conjunto de datos grande y representativo, que suele denominarse *conjunto de entrenamiento*. A continuación, dado un e-mail de test x, se calculará su distribución de probabilidad h_x y se medirá la distancia entre h_x y la distribución de cada clase c, h_c: $d(h_x, h_c)$.

Habitualmente, se asigna a x la clase que minimiza la distancia:

$$c^*(x) = \arg \min_{c=1,\ldots,C} d(h_x, h_c) \tag{3}$$

y se considera que se ha producido un error cuando $c^*(x)$ no coincide con la clase $c_{\text{true}}(x)$ a la que pertenece *realmente* el e-mail. Sin embargo, esta definición del error no informa sobre cómo de cerca hemos estado de acertar la clase de cada e-mail. Para una mejor caracterización del rendimiento del método de clasificación, considerese el conjunto $C_K(x)$, formado por las K clases más próximas al e-mail x según dicho método. Evidentemente, $C_1(x) = \{c^*(x)\}$. Si la clase a la que realmente pertenece el e-mail x está entre las K más próximas, consideraremos que el método ha acertado. Por tanto, se producirá un error de clasificación sólo si $c_{\text{true}}(x) \notin C_K(x)$.

El programa deberá calcular la tasa de error que se obtiene al clasificar todos los e-mails de test. Si n_K es el número de errores de clasificación que se producen aplicando el criterio que acabamos de describir (para un cierto valor de K), y N es el número total de e-mails de test, la tasa de error (en porcentaje) se calculará como:

$$\%Error@K = n_K \cdot 100 \ / \ N \tag{4}$$

Se calculará y presentará la tasa de error $\%Error@K$ para $K = 1, \ldots, 5$. Nótese que $\%Error@1$ representa el error de clasificación habitual.

Podrá experimentarse con distintas definiciones de la función distancia. Considerense un e-mail x y una clase c, y sean $h_x(w)$ y $h_c(w)$ las frecuencias de una palabra w en el e-mail x y en la clase c, respectivamente. Se propone usar las distancias siguientes:

- **Distancia de superposición:**

$$d(h_x, h_c) = 1 - \sum_{\forall w \in x} \text{mín}(h_x(w), h_c(w)) \tag{5}$$

- **Distancia euclídea:**

$$d(h_x, h_c) = \sqrt{\sum_{\forall w \in \cup\{x,c\}} \left(h_x(w) - h_c(w)\right)^2} \tag{6}$$

- **Distancia de correlación:**

$$d(h_x, h_c) = 1 - \sum_{\forall w \in x} h_x(w) \cdot h_c(w) \tag{7}$$

Se dispone del conjunto de datos *20news-bydate*, que contiene 20 clases de e-mails (en inglés), definidas según su temática (los e-mails han sido extraídos de diferentes grupos de noticias). Para cada clase de e-mails se dispone de dos subconjuntos: (1) *train* (60 % del total), utilizado para estimar las distribuciones de probabilidad de las clases; y (2) *test* (40 % del total), utilizado para medir el error de clasificación. El criterio utilizado para definir ambos subconjuntos ha sido la fecha de envío de los e-mails (de ahí la extensión *bydate*), de manera que todos los e-mails de *test* son posteriores a los de *train*. Ello garantiza una mayor independencia entre unos y otros.

Se suministran listas para cada una de las clases de los subconjuntos *train* y *test*. El conjunto de datos *20news-bydate* junto con las listas de e-mails se pueden descargar desde cualquiera de los siguientes enlaces (con descargar y descomprimir uno de ellos es suficiente):

http://www.ehu.eus/~ljrf/tmp/20news-bydate.tgz
http://www.ehu.eus/~ljrf/tmp/20news-bydate.zip

Sugerencia: La presencia en los e-mails de numerosas palabras comunes (artículos, preposiciones, ciertas formas verbales, etc.) hace que la clasificación sea difícil. Asimismo, hay también palabras que aparecen muy pocas veces. En particular, aquellas que solo aparecen una vez en el conjunto de entrenamiento apenas aportan información. Se propone eliminar de las distribuciones tanto las palabras muy comunes como las palabras muy raras. Ello debería redundar en una mejora muy notable del rendimiento.