S10 - Grafos Probabilisticos y Analisis de Textos

Juan Carlos Martinez-Ovando *ITAM*

Revisaremos estructuras probabilisticas que pueden definirse al rededor de objetos `grafos`. Revisaremos aspectos inferenciales relacionados con el aprendizaje estadistico en estructuras de `grafos probabilisticos`. Estudiaremos el modelo grafico probabilistico *naive Bayes* para clasificacion supervisada. Revisamos los fundamentos del analisis de textos usando las librerias `tm` y `SnowballC`.

Keywords: Graphical models, text analytics, contingency tables.

Paquetes de R

Aplicaremos el modelo **naive Bayes** en el contexto de analisis y clasificacion de textos usando la libreria `e1071`

```
if(!require('tm')){install.packages("tm")}
```

```
## Loading required package: tm
```

```
## Loading required package: NLP
```

```
if(!require('tm')){install.packages("SnowballC")}
if(!require('tm')){install.packages("e1071")}
```

Nota: Este `markdown` ha sido producido con `R-3.3.3`.

Grafos probabilisticos

Hemos visto que los `grafos` son herramientas intuitivas para la visualizacion de relaciones entre variables (ya sea a nivel `objeto` o entre `objetos`). En los `grafos` hemos identificado:

- Cada `nodo` representa una variable (i.e. medicion particular de un objeto o el objeto mismo)
- La relacion entre `nodos` esta representada por `ligas` de conexion
- Las `ligas` de conexion pueden indicar `relaciones simetricas` o `relaciones asimetricas`, dependiendo de la direccion de las `ligas`

Un `grafo probabilistico` emplea las relaciones de un `grafo` normal e introduce en las `ligas de conexion` una **medida de probabilidad** para cuantificar el grado de conectividad entre valores especificos de los nodos

Asi, las **relaciones probabilisticas** entre los valores especificos de dos `nodos` $\{i, j\}$, donde los valores especificos son X_i y X_j se definen como:

a. $\mathbb{P}(X_i, X_j) = \mathbb{P}(X_i|X_j)\mathbb{P}(X_j) = \mathbb{P}(X_j|X_i)\mathbb{P}(X_i) =$ si la relacion es **simetrica**

b. $\mathbb{P}(X_i, X_j) = \mathbb{P}(X_i|X_j)\mathbb{P}(X_j)$ si la relacion es **asimetrica**

$$X_j \to X_i$$

c. $\mathbb{P}(X_i, X_j) = \mathbb{P}(X_j|X_i)\mathbb{P}(X_i)$ si la relacion es **asimetrica**

$$X_i \to X_j$$

Una vez definidas estas relaciones probabilisticas, usualmente imponemos una estructura parametral de la siguiente forma:

$$\mathbb{P}(X_i, X_j) = F(X_i, X_j|\theta_{ij}, \gamma),$$

donde

- θ_{ij} es un parametro `local` para los nodos conectados `(i,j)`
- γ es un parametro general para todos los `nodos`
- F es una funcion de probabilidad

La direccionalidad de la relacion esta asociada con la forma funcional de `F`, que es una **funcion de distribuciones de probabilidades**.

Independencia condicional

Cuando un grafo cuenta con un conjunto de nodos, $\{1, \ldots, n\}$ para los cuales se tienen asociados valores especificos, $\{X_1, \ldots, X_n\}$, podemos pensar en estructuras de **dependencia estocastica** entre conjuntos de `variables` o `nodos`, i.e. para todo $i = 1, \ldots, n$ tenemos que

$$\mathbb{P}(X_i|X_{-i}) = F(X_i|X_{\delta(i)}, \theta_i, \gamma),$$

donde

- X_{-i} denota el conjunto de variables en el grafo con excepcion de la i-esima variable
- $X_{\delta(i)}$ denota un conjunto de variables asociado con el conjunto de `nodos` conectados con el i-esimo (i.e. los `vecinos` de la i-esima coordenada o `nodo` del grafo)
- θ_i es el parametro local
- γ parametro global

Al rededor de la estructura anterior podemos pensar en la nocion de **independencia estocastica** entre dos `nodos` $\{i, j\}$ para sus variables X_i y X_j, condicional en la informacionm compartida para ambas, dada por sus vecindades $X_{\delta(i,j)}$ **si y solo si**

$$F(X_i, X_j|X_{\delta(i,j)}, \theta_{i,j}, \gamma) = F(X_i|X_{\delta(i,j)}, \theta_{i,j}, \gamma) \times F(X_j|X_{\delta(i,j)}, \theta_{i,j}, \gamma).$$

La **independencia estocastica marginal** entre X_i y X_j se obtiene cuando la anterior relacion se cumple con $X_{\delta(i,j)} = \emptyset$.

Tipos de grafos probabilisticos

A. Grafos no dirigidos

B. Grafos dirigidos

C. Grafos de factores

Referencias

- **Bishop** - Pattern Recognition and Machine Learning (capitulo 3)
- **Jordan & Weiss** - Probabilistic Inference in Graphical Models
- **Heckerman** - Graphical Models: Structure Learning

Naive Bayes

El modelo **naive Bayes** se emplea para relaizar tareas de clasificacion supervizada. Recordemos que un `clasificador` es una funcion

$$f : (x_1, \ldots, x_p) \to y,$$

donde

- $x_1, \ldots, x_p$ son `atributos`
- y es una `etiqueta` (tipicamente binaria)

De acuerdo a lo que vimos antes, si $\mathbb{P}$ es una regla de probabilidad asociada con el clasificador f tenemos que el clasificador bayesiano optimo es aquel

$$\hat{y} = \arg\max_y \widehat{\mathbb{P}}\left(y|x_1, \ldots, x_p\right),$$

donde $\widehat{\mathbb{P}}$ es una estimacion de $\mathbb{P}$, pues la ultima es desconocida.

La especificacion de $\mathbb{P}\left(y|x_1, \ldots, x_p\right)$ puede obtenerse via el Teorema de Bayes, reconociendo que los atributos son aleatorios, i.e.

$$\mathbb{P}\left(y|x_1, \ldots, x_p\right) = \frac{\mathbb{P}\left(x_1, \ldots, x_p|\, y\right)\mathbb{P}(y)}{\mathbb{P}\left(x_1, \ldots, x_p\right)}.$$

Cuando los atributos son continuos se require de un gran ejercicio de abstraccion para simplificar lo que se conoce como la `verosimilitud` del modelo, i.e. $\mathbb{P}\left(x_1, \ldots, x_p|\, y\right)$. *El uso de modelos graficos probabilisticos nos permite hacer esto, hasta ciertos limites.*

Pensemos que los p-`atributos` son binarios y que las `etiquetas` contienen K categorias. En este caso, el numero efectivo de parametros necesarios para especificar el clasificador probabilistico seria igual a $K(2^p - 1)$, el cual puede ser siginificativamente grande e intratable en varios contextos (esto se conoce como `curse of dimensionality`).

Grafos probabilisticos

Empleando los grafos probabilisticos, podemos invocar (imponer, en realidad) ciertas estructuras de independencia condicional. En particular, el **supuesto fundamental** del modelo **naive Bayes** cuando y son etiquetas observables y $x_1, \ldots, x_p$ son p-atributos binarios es que los atributos x_js son condicionalmente independientes dada la etiqueta y.

Bajo el supuesto anterior, el numero efectivo de parametros del modelo se reduce de $K(2^p - 1)$ a pK, dentro del contexto mencionado.

Asi, el clasificador bayesiano **naive** queda definido como

$$\widehat{y_{NB}} = \arg\max_y \widehat{\mathbb{P}}(y) \prod_{j=1}^{p} \widehat{\mathbb{P}}(x_j|y).$$

En este caso:

- El componente $\mathbb{P}(y)$ se conoce como la `prior` para la clasificacion
- Los atributos x_j son condicionalmente independientes dado y, con *verosimilitud* dada por $\mathbb{P}(x_j|y)$.

Aprendizaje estadistico

El aprendizaje estadistico en la clase de modelos **naive Bayes** se realiza mediante un procedimiento de actualizacion de distribuciones multinomiales con conteos de frecuencias por clases, i.e.

$$\begin{aligned} \widehat{\mathbb{P}}(y = k) &= \frac{\#\{y_i : y_i = k\}}{n}, \\ \widehat{\mathbb{P}}(x_j = l|y = k) &= \frac{\#\{(y_i, x_{ij}) : y_i = k, x_{ij} = l\}}{\#\{y_i : y_i = k\}}, \end{aligned} \tag{1}$$

donde n es el tamano de la muestra.

El clasificador **naive Bayes** se emplea con mucho exito en problemas de **analisis de textos** porque:

- En esos casos los atributos x_j representan la aparicion (mas no la frecuencia ni el orden) de ciertos `terminos`.
- El numero de `terminos` o p es significativamente grande, por lo que la simplificacion del supuesto fundamental del **naive Bayes** resulta muy practica.

Datos de textos

Cargamos los datos `aclImdb` correspondiente a la base de datos *Large Movie Review Dataset* de IMDB, con 25 mil resenas en la `muestra de entrenamiento` y 25 mil registros en la `muestra de prueba`. Los datos los pueden descargar de la siguiente Liga.

Los registros corresponden a textos cortos con resenas sobre peliculas de IMDB. Los datos comprenden tanto las resenas como los puntajes de los criticos.

***Definicion*:** En linguistica, el **Corpus** se define como una coleccion de documentos (una especie de `data.frame` como ya conocemos, pero donde los registros son documentos). La libreria `tm` permite definir esta clase de objetos en `R`, reconociendo que los documentos son cadenas (`strings`) de caracteres.

Cargamos y definimos el `corpus` con las siguientes intrucciones:

```
rm(list=ls())

# librerias
require("tm")
require("SnowballC")

load("C:/Users/jcmo/Google Drive/Material.Cursos/EST25134/Sessions/IMDBReviews_Data.Rdata")

# creacion del ‘corpus‘
nb_pos <- VCorpus(DirSource("/home/jcmo/Naive.Bayes/aclImdb/train/pos"),
                  readerControl = list(language="en"))

nb_neg <- VCorpus(DirSource("/home/jcmo/Naive.Bayes/aclImdb/train/neg"),
                  readerControl = list(language="en"))

class(nb_pos); class(nb_neg)
```

- La creacion del `corpus` requiere de la identificacion del idioma de los textos.La libreria `SnowballC` nos permite indicar esto. (En el ejemplo, seleccionamos la opcion `en`, revisen la documentacion de la libreria para revisar los otros idiomas disponibles, incluyendo al espanol).

Operaciones sobre `corpus`

La fusion de dos `corpus` en un solo objeto puede obtenerse empleando la instruccion `c()` –como en arreglos de datos en `R`–, solo que en este caso la opcion `recursve=T` es necesaria para mantener la estructura de `corpus`.

```
nb_all <- c(nb_pos, nb_neg, recursive=T)
class(nb_all)
```

Las entradas del objeto `corpus` pueden accederse como si fueran objetos del tipo `list`, pero con metadatos particulares, e.g.:

```
nb_all[[1]]
class(nb_all[[1]])
meta(nb_all[[1]])
```

Creamos ahora un vector con los nombres de los archivos:

```
ids <- sapply(1:length(nb_all), function(x) meta(nb_all[[x]], "id"))
head(ids)
```

Cada archivo es de la forma `<counter>_<score>.txt` y tienen asociado puntajes.

- Puntajes en el rango de 7-10 son etiquetados como positivos (`pos`)
- Puntajes en el rango de 0-4 son etiquetados como negativos (`neg`)

Extraemos los puntajes/calificaciones de los archivos empleando las siguientes funciones:

```
scores <- as.numeric(sapply(ids,
                            function(x) sub("[0-9]+_([0-9]+)\\.txt", "\\1", x)))
scores <- factor(ifelse(scores>=7,"positive","negative"))
summary(scores)
```

Algunas instrucciones de preprocesamiento, empleando la funcion `tm_map`, e.g.:

- Eliminar numeros
- Eliminar puntuacion
- Convertir mayusculas en minusculas
- Eliminar palabras
- Eliminar espacios extras entre palabras

```
nb_all <- tm_map(nb_all, content_transformer(removeNumbers))
nb_all <- tm_map(nb_all, content_transformer(removePunctuation))
nb_all <- tm_map(nb_all, content_transformer(tolower))
nb_all <- tm_map(nb_all, content_transformer(removeWords), stopwords("english"))
nb_all <- tm_map(nb_all, content_transformer(stripWhitespace))
```

Creamos ahora la matriz de terminos de los documentos:

```
nb_dtm <- DocumentTermMatrix(nb_all)
dim(nb_dtm)
class(nb_dtm)
nb_dtm[12,12]
nb_dtm[1,1]
nb_dtm
```

Eliminamos terminos que son poco frecuentes (`sparse`):

```
nb_dtm <- removeSparseTerms(x=nb_dtm, sparse = 0.90)
dim(nb_dtm)
nb_dtm[12,12]
nb_dtm[1,1]
```

Exploramos la primera resena de peliculas, identificando los terminos que estan relacionados (i.e. los que tienen frecuencia distinta de `0`):

```
inspect(nb_dtm[1,])
terms <- which( inspect(nb_dtm[1,]) != 0 )
inspect( nb_dtm[1,terms] )
```

Convertimos ahora todos los elementos del objeto `nb_dtm` en entradas binarias, pues el analisis de textos campleta si el termino especifico (e.g. `superv`) aparece en el texto, mas no toma en cuenta *cuantas* veces aparece en el texto.

```
nb_dtm <- weightBin(nb_dtm)
inspect( nb_dtm[1,terms] )
```

Dividimos el `corpus` en datos `train` y `test` en formato `data.frame`. La segmentacion de `nb_dtm` se realiza de manera aleatorizada:

```
nb_df <- as.data.frame(as.matrix(nb_dtm))
set.seed(1)
nb_sampling_vector <- sample(25000, 20000)
nb_df_train <- nb_df[nb_sampling_vector,]
nb_df_test <- nb_df[-nb_sampling_vector,]
scores_train <- scores[nb_sampling_vector]
scores_test <- scores[-nb_sampling_vector]

save(nb_df_train,
     nb_df_test,
     scores_train,
     scores_test,
     file = "IMDBReviews_Data.Rdata")
```

Ilustracion

La implementacion del modelo **naive Bayes** la realizamos empleando la libreria `e1071` y los datos de entrenamiento `nb_df_train`:

```
library("e1071")

load("C:/Users/jcmo/Google Drive/Material.Cursos/EST25134/Sessions/IMDBReviews_Data.Rdata")

nb_model <- naiveBayes(nb_df_train, scores_train)

summary(nb_model)
```

Calculamos las predicciones `ajustadas` empleando el modelo estimado `nb_model` sobre los datos de entrenamiento `nb_df_train`:

```
nb_train_predictions <- predict(nb_model, nb_df_train)
save(nb_df_train,
     nb_df_test,
     scores_train,
     scores_test,
     nb_train_predictions,
     file = "IMDBReviews_Data.Rdata")
```

```
# errores ajustados de clasificacion
mean(nb_train_predictions == scores_train)
table(actual = scores_train, predictions = nb_train_predictions)
```

Ahora, calculamos las predicciones `reales` empleando el modelo estimado `nb_model` sobre los datos de prueba `nb_df_test`:

```
nb_test_predictions <- predict(nb_model, nb_df_test)

# errores de prediccion
mean(nb_test_predictions == scores_test)
table(actual = scores_test,
      predictions = nb_test_predictions)
```

Post-procesamiento de los datos (`stem`):

```
nb_all <- tm_map(nb_all, stemDocument, language = "english")
nb_dtm <- DocumentTermMatrix(nb_all)
nb_dtm <- removeSparseTerms(x=nb_dtm, sparse = 0.99)
nb_dtm <- weightBin(nb_dtm)
nb_df <- as.data.frame(as.matrix(nb_dtm))
nb_df_train <- nb_df[nb_sampling_vector,]
nb_df_test <- nb_df[-nb_sampling_vector,]
```

Contraste del modelo con los `corpus` post-procesados:

```
nb_model_stem <- naiveBayes(nb_df_train, scores_train)

nb_test_predictions_stem <- predict(nb_model_stem, nb_df_test)

mean(nb_test_predictions_stem == scores_test)
table(actual = scores_test,
      predictions = nb_test_predictions_stem)
```

Nota: Calcula de nuevo `nb_dtm` sin haber realizado el post-procesamiento de los datos antes de continuar con las siguientes instrucciones.

```
nb_all <- c(nb_pos, nb_neg, recursive=T)
nb_all <- tm_map(nb_all, content_transformer(removeNumbers))
nb_all <- tm_map(nb_all, content_transformer(removePunctuation))
nb_all <- tm_map(nb_all, content_transformer(tolower))
nb_all <- tm_map(nb_all, content_transformer(removeWords),
                 stopwords("english"))
nb_all <- tm_map(nb_all, content_transformer(stripWhitespace))
nb_dtm <- DocumentTermMatrix(nb_all)
nb_dtm <- removeSparseTerms(x=nb_dtm, sparse = 0.99)
nb_df <- as.data.frame(as.matrix(nb_dtm))
nb_df_train <- nb_df[nb_sampling_vector,]
nb_df_test <- nb_df[-nb_sampling_vector,]
```

```
nb_model_laplace <- naiveBayes(nb_df_train, scores_train, laplace=10)

nb_test_predictions_laplace <- predict(nb_model_laplace, nb_df_test)

mean(nb_test_predictions_laplace == scores_test)
table(actual = scores_test,
      predictions = nb_test_predictions_laplace)
```

Referencias adicionales

- **Maas** - *Learning Word Vectors for Sentiment Analysis*
- **Feinerer et al** - *Text mining infrastructure in R*