DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Redes Neuronales

Trabajo práctico 1

Parser

Resumen

Entrenamiento de Redes Neuronales

Integrante	LU	Correo electrónico
Negri, Franco	893/13	franconegri2004@hotmail.com
Rey, Martin Gaston Podavin	483/12	marto.rey2006@gmail.com

Palabras claves:

TP

Índice

1. Introduccion

En este trabajo practico nos propondremos clasificar mails en spam como en no spam (conocido tambien como ham).

Para esto analizaremos los datos en busca de atributos utiles y luego experimentaremos con diversos clasificadores con la intención de encontrar los que mejor clasifiquen.

Ademas utilizaremos tecnicas de reduccion de dimencionalidad con la intención de mejorar aun mas los modelos de los puntos anteriores.

Para tener una buena metrica de los resultados obtenidos, apartamos parte del set de datos que nos fue entregado para ser utilzado al final de la experimentación y asi tener una visión realista de que tan buenos son los clasificadores elegidos. Con el resto de los datos, realizamos kfolds con k igual a diez con la intención de minimizar en cierta medida el overfiting de datos.

2. Desarrollo

Como ya adelantamos en la seccion anterior el objetivo de este tp será clasificar mails en spam y no spam, para ellos, comenzaremos el trabajo seleccionando atributos adecuados que concideramos dividen bien el problema.

2.1. Selección De atributos

Para la selección de atributos observamos parte del json entregado en busca de palabras claves que pudieran distinguir entre spam y no spam (desde ahora, ham).

De este analisis se encontró que palabras tales como *viagra* o *nigeria* son muy frecuentes en los mensajes de spam, asi tambien aquellas que hagan referencia a negocios o a dinero, por lo que incluimos atributos que cuenten las veces que son mencionados estas palabras en el cuerpo y el encabezado del mensaje. Así tambien observamos que los mails de spam tienden a tener enlaces hacia sitios web o contenido html en el cuerpo del mensaje por lo que tambien contavilizamos la cantidad de ocurrencias de palabras tales como html o http y simbolos especiales como la barra invertida, o el hashtag.

De esta menera reunimos al rededor de 100 atributos que utilizaremos a continuación para la experimentación.

2.2. Experimentación

Los clasificadores que utilizaremos para experimentar en este trabajo serán:

- Arboles de Deciciones
- Naive bayes Multinomial
- Vecinos mas Cercanos
- SVC
- Random Forest

Para cada uno de ellos utilizaremos grid search para intentar encontrar los hiperparametros que logren la mejor clasificación.

En cada una de las experimentaciónes utilizamos grid search de sklearn con 10 kfolds.

2.2.1. Arboles de Deciciones

Como vimos en clase, este clasificador intentará aplicar una serie de reglas sucesivas para determinar la clasificación. Por ejemplo, un posible arbol de decición para clasificar spam podría ser, si el mensaje contiene la palabra *nigeria* mas de 4 veces, es spam, si no ver cuantas veces se menciona *html* en el mensaje, si el resultado esta entre 4 y 7 es spam, si no no... y así sucecivamente.

En particular, los hiperparametros que querremos encontrar para este algoritmo en particular son: la maxima profunididad del arbol, y la cantidad minima de muestras necesarias para dividir un nodo.

Experimentamos variando los hiperparametros entre los siguientes valores:

- $max\_depth : 1, 3, 5, 10, 15, 50, 100$
- $min\_samples\_split : 1, 3, 5, 10, 15$

El mejor resultado obtenido fue de 0,965975308642 con un max_depth de 50 y un min sammple split de 1.

2.2.2. Naive bayes Multinomial

El objetivo del Naive bayes será encontrar la clase mas probable de cada instancia en base al calculo de la probabilidad bayesiana.

Para este clasificador querremos encontrar el mejor suabizado laplaciano ($alfa$), para ello experimentamos con los siguientes valores:

- $alfa : 0{,}1, 0{,}2, 0{,}3, 0{,}4, 0{,}5, 0{,}6, 0{,}7, 0{,}8, 0{,}8, 0{,}9, 1$

El resultado del grid search arroja que el mejor score resulto ser de 0,587740740741 con un alpha de 0,1

2.2.3. Vecinos mas Cercanos

La idea tras este clasificador es dada una nueva instancia a clasificar, comparar sus atributos contra todos los de la base de entrenamiento y quedarnos con k vacinos mas *cercanos*. La *cercania* de una instancia con otra puede calcularse como la norma dos de los atributos al cuadrado o alguna otra función que nos permita definir una distancia relativa entre dos hiperplanos. Para este clasificador querremos determinar la cantidad de vecinos optimo y la función de peso a utilizar en la predicción. Para esto ultimo tendremos dos posivilidades, con pesos uniformes o por distancia. Para poner un ejemplo sobre cual es la diferencia, supongo que seteo como hiperparametro que la cantidad de vecinos es igual a 3 y para un mensaje dado obtengo que el primer vecino es spam y los otros dos son ham. Con una función uniforme, se determinará que el mail es ham, ya que ham es mayoría. Pero podría darse el caso en que el primer vecino se encuentre a una distancia infima del mail a clasificar y que los otros dos se encuentren a una distancia extremadamente grande. En ese caso sería mas razonable asignarle mas peso al primer vecino de manera tal que el mail sea clasificado como spam. Esta segunda solución será la de utlizar pesos variables que dependan de la distancia.

Para este clasificador provamos con $1, 3, 5, 7, 10$ y 15 vecinos y el mejor resultado obtenido fue de 0,908358024691 con una cantidad de vecinos igual a 1 y pesos uniformes.

2.2.4. SVC

2.2.5. Random Forest

Este clasificador consiste en crear una gran cantidad de arboles de decición, y al momento de clasificar elegir el resultado que sea la moda de las clasificaciones determinadas por los arboles individuales.

Para este estimador querremos definir la cantidad de arboles de decición a utilizar, la profundidad maxima que se le permitirá tener a cada arbol, y maxima cantidad de atributos a conciderar cuando se este realizando la división de un nodo.

Experimentamos con $2, 5, 10, 15, 40, 100$ arboles, $2, 5, 10, 20$ atributos a considerar al momento de la división de un nodo y una profundidad maxima de $3, 5, 20$ y sin restricciones

Para este estimador el mejor resultado fue de 0,979098765432 con una cantidad de arboles igual a 100, una profundidad irrestricta y una cantidad de features a examinar igual a 10.

2.3. PCA

Para intentar mejorar la precición de los resultados utilizamos PCA para obtener las componentes principales de nuestro set de atrubutos.

Ademas utlizamos grid search para obtener cual era la cantidad optima de componentes principales para cada uno de los algoritmos. Utilizamos $2, 5, 10, 40, 70$ componentes para la experimentación.

Lo que observamos fue que tanto en arboles de decición como en random forests las respuestas resultaron ser levemente peores, pasando de un mejor resultado de 0,965975308642 a 0,931592592593 y de 0,967962962963 a 0,979098765432, respectivamente.

Para el algoritmo de naive vayes esta vez utilizamos un modelo gaussiano, pero los resultados siguieron sin ser muy buenos, obteniendo un score del 0,596308641975

Para vecinos mas cercanos el cambio de performance no fue notable, obteniendo una respuesta de 0.908.

Si bien entendemos que los resultados del grid search podrian haber sido mejores variando nuevamente los parametros del apartado anterior junto con la cantidad de componentes (sobretodo en el caso de vecinos mas cercanos), esto también resultaba muy caro computacionalmente y por cuestiones de tiempo no lo incluimos en el trabajo.

3. Concluciones