Aprendizaje Automatico TP2: Paper Nro. 2 - Multi-label Learning Algorithms

Lebrero, Ignacio — Alejandro, Gonzalez — Mauro, Cherubini

Diciembre 2018

1 Introducción

Los algoritmos de clasificacion tal cual los venimos viendo en la materia consideran la hipotesis de que cada objeto del mundo real corresponde a un solo concepto (al que representamos como una clase). Esto no necesariamente es verdad, cosas como *que idiomas habla, que nacionalidades tiene o cuales ingredientes lo conforman* son ejemplos de clasificadores que no encajan del todo bien con esta hipotesis, pues corresponden no solo a uno, si no a un conjunto de coneceptos. De esto se desprende la consideración de problemas y datasets *multi-label*, al cual cada instacia x_i le corresponde un conjunto de etiquetas Y_i.

2 Metricas de Evaluacion

A la hora de estimar el *score* de un clasificador multi-label existen muchas metricas posibles en las que nos podemos basar, a continuación expondremos algunas a modo de ejemplo.

Antes que nada definiremos un poco de notación que será necesaria para las subsiguientes definiciones. Notamos a [[.]] como la operación que toma un predicado y devuelve 1 si este se cumple, y 0 en caso contrario. Y notamos a Δ como la diferencia simetrica entre dos conjuntos.

Sea h un clasificador multi-label, $S = \{(x_i, Y_i) \mid 1 \leq i \leq p\}$ el conjunto de validación, y q la cantidad de etiquetas existentes, podemos definir como posibles metricas, entre otras:

$$subsetacc(h) = \frac{1}{p}\sum_{i=1}^{p} [[h(x_i) = Y_i]] \qquad hloss(h) = \frac{1}{p}\sum_{i=1}^{p} \frac{1}{q} |h(x_i)\Delta Y_i|$$

3 Métodos de transformación de problemas

En este enfoque la idea es llevar el problema de tener multiples etiquetas a otros escenarios conocidos en donde sea mas fácil tratarlos, por lo que el foco de estudio se encuentra en hallar metodos prácticos que permitan lograr esto de manera eficiente.

3.0.1 Relevancia binaria

La idea basica del algoritmo es descomponer el problema de aprendizaje multiclase en q problemas de clasificacion binaria independientes (Cada problema de clasificacion corresponde a una posible etiqueta en el espacio de la etiqueta). Para cada y_j (etiqueta de la *j-clase*) primero se construye el correspondiente conjunto de entrenamiento binario considerando la relevancia de cada ejemplo de entrenamiento como:

$$D_j = \{(x_j, \phi(Y_j, y_j)) \mid 1 \leq i \leq m\} \quad \text{donde} \quad \phi(Y, y) = \begin{cases} +1 & y \in Y \\ -1 & \text{caso contrario} \end{cases}$$

Dado esto por medio de algún algoritmo de clasificacion binaria generamos q clasificadores binarios $g_j(.)$, con $1 \leq j \leq q$. Luego para poder predecir la clase Y correspondiente a una nueva instancia x no vista, combinamos los q clasficadores para predecir en función de la relevancia predicha si la etiqueta correspondiente debería formar parte del conjunto Y predicho, formalmente lo expresamos como Y = $\{y_i | g_j(x) > 0$,

$1 \leq j \leq q\}$ De acuerdo a esta definición si todos los clasificadores nos dan una respuesta negativa el conjunto predicho Y resulta ser el conjunto vacio. Para evitar esta indeseada consecuencia utilizamos la regla conocida como $T-Criterion$ de manera de complementar con las etiquetas de mayor relavancia, con lo cual formalmente definimos al Y predicho como $Y = \{y_i | g_j(x) > 0,\ 1 \leq j \leq q\} \cup \{y_k |\ k = argmax_{1 \leq k \leq q}\ g_k(x)\}$

3.0.2 Cadenas de clasificador

La idea detras de este algoritmo es transformar el problema de aprendisaje multinivel en una cadena de problemas de clasificacion binaria, donde la subsecuencia de clasificadores binarios en la cadena se basan en las predicciones de los anteriores.
Para las q posibles clases de etiquetas $\{y_1, y_2, ..., y_q\}$, se define una función de permutación τ: $\{1..q\} \rightarrow \{1..q\}$ que se usa para especificar un orden $(y_{\tau(1)} \succ y_{\tau(2)} \succ .. \succ y_{\tau(q)})$. En base a esta función definimos un *dataset* asociado a cada valor asignado $\tau(j)$ para $1 \leq j \leq q$.

$$D_{\tau(j)} = \{\ ([x_i, {pre^i}_{\tau(j)}]\ , \phi(Y_i, y_{\tau(j)}))| i \leq i \leq m\} \quad \text{donde} \quad {pre^i}_{\tau(j)} = (\phi(Y_i, y_{\tau(1)}), .., \phi(Y_i, y_{\tau(j-1)}))^T$$

Luego en base a algún algoritmo de aprendizaje binario B inducimos un clasificador binario $g_{\tau(j)} : Xx\{-1, +1\}^{j-1} \rightarrow \mathbb{R}$ de la forma $B(D_{jk}) \rightarrow g_{jk}$, que determina la relevancia de $y_{\tau(j)}$.

Para instancias x aún no vistas, su conjunto de etiquetas predichas estará definido como $Y = \{y_{\tau(j)} | {\lambda^x}_{\tau(j)} = +1,\ 1 \leq j \leq q\}$, donde definimos a los $\lambda^x_{\tau(j)} \in \{-1, +1\}$, que representan la relevancia binaria de $y_{\tau(j)}$ en x, como:

$\lambda^x_{\tau(1)} = sign[g_{\tau(1)}(x)]$

$\lambda^x_{\tau(j)} = sign[g_{\tau(j)}([x, \lambda^x_{\tau(1)}, ..., \lambda^x_{\tau(j-1)}])] \quad (1 \leq j \leq q)$

3.0.3 Random k-Labelsets

Esta técnica consiste en transformar nuestro problema en el cual consideramos para cada x_i del conjunto de entrenamiento un conjunto de etiquetas Y_i asociado; a un problema de clasificación multi-clase simple, en donde a cada x_i se lo asocia con un $n_i \in \mathbb{N}$. Para esto debemos seleccionar una función inyectiva $\sigma_Y : 2^Y \rightarrow \mathbb{N}$ que nos permita mapear subconjuntos de Y a números naturales.

Debido a que trabajar directamente con los Y_i tal como vienen dados en el dataset resulta inpractico en terminos de complejidad y generalidad, lo que hacemos es elegir aleatoriamente n subconjuntos $Y^k(l) \in Y^k$ con $1 \leq l \leq n \leq \binom{q}{k}$ y $q = |Y|$, donde Y^k es el conjunto de subconjuntos de tamaño k de Y. Luego a partir de esta selección generamos n conjuntos de entrenamiento distintos asociados cada uno a un $Y^k(l)$, y utilizamos a estos últimos como base para generar un ensamble de n componentes. Cada uno de estos nuevos conjuntos de entrenamiento se los define formalmente como:

$${D^\dagger}_{Y^k(l)} = \{(x_i, \sigma_{Y^k(l)}(Y_i \cap Y^k(l)))| 1 \leq i \leq m\}$$

Donde entonces el nuevo conjunto de clases asociado a cada ${D^\dagger}_{Y^k(l)}$ será $\Gamma({D^\dagger}_{Y^k(l)}) = \{\sigma_{Y^k(l)}(Y_i \cap Y^k(l))| 1 \leq i \leq m\}$ Además a cada componente del ensamble se le corresponde un clasificador multi-clase simple ${g^\dagger}_{Y^k(l)}(.)$, el cual es entrenado solo en base a su respectivo ${D^\dagger}_{Y^k(l)}$.

Con este ensamble entonces para clasificar alguna nueva instancia x calculamos el correspondiente conjunto de etiquetas Y como:

$$Y = \{y_j \mid (\mu(x, y_j)/\tau(x, y_j)) > 0,5\ ,\ 1 \leq j \leq q\}$$

Donde $\mu(x, y_j) = \sum_{r=1}^{n}[[y_j \in \sigma^{-1}_{Y^k(l)}({g^\dagger}_{Y^k(l)}(x))]]$ con $1 \leq j \leq q$, representa la cantidad de votos que el ensamble le da a y_j en favor a si esta etiqueta deberia pertenecer o no al conjunto Y. Mientras que $\tau(x, y_j) = \sum_{r=1}^{n}[[y_j \in Y^k_{l_r}]]$ con $1 \leq j \leq q$, representa la cantidad maxima de estos votos que se le podría llegar a asignar a y_j. Con estas dos metricas presentes observemos que el conjunto Y asociado a x se entiende como todas aquellas eiquetas y_j cuya valoración en el ensamble (con relación a x) es major a la mitad de la valoración maxima que se le podría llegar a dar.

4 Métodos de Adaptación de algoritmos

Este enfoque a diferencia del anterior consiste en el estudio de metodos que logren adaptar algoritmos ya conocidos para que puedan trabajar directamente con datasets multi-label sin necesidad de tener que transformarlos.

4.0.1 Multi-Label k-Nearest Neighbor (ML-kNN)

La idea basica de este algoritmo es adaptar la tecnica de k vecinos mas cercanos para lidear con el problema de multi-label, utilizando para ello la regla de *maximun a posteriori* (MAP) para realizar predicciones con la información de las etiquetas que poseen sus vecinos.

Sea C_j el numero de los vecinos de x con la etiqueta y_j, y sea H_j el evento en el que x tiene la etiqueta y_j, entonces $P(H_j|C_j)$ representa la probabilidad a posteriori de que H_j mantenga la condición de que x contiene C_j vecinos con la etiqueta y_j. Analogamente $P(\neg H_j|C_j)$ representa la probabilidad a posteriori de que H_j no mantenga esa condicion. Con estas estimaciones de acuerdo a la regla MAP el conjunto predicho para alguna instancia aún no vista x queda determinado como $Y = \{y_j \mid P(H_j|C_j)/P(\neg H_j|C_j) > 1,\ 1 \leq j \leq q\}$, recordar que C_j se define en función de la instancia x.

Para calcular el cociente expresado en la definición de Y, se lo reduce por medio del teorema de Bayes a una forma que depende solo de $P(H_j)$, $P(\neg H_j)$, $P(C_j|H_j)$, y $P(\neg C_j|H_j))$, las cuales luego son estimadas por medio de estrategias de *frecuency counting*.

4.0.2 Multi-Label Decision Tree (ML-TD)

La idea de este algoritmo es adaptar la tecnica del *arbol de decision* a multi-labeling. Donde la principal componente de la tecnica sera el uso de entropia multi-label (MLEnt). Para simplificar las cosas asumiremos independencia entre los labels y defniremos T como un dataset multi-label con n samples definido con la misma forma que D previamente.

Primero definimos la entropia multilabel como:

$$\text{MLEnt(T)} = \sum_{i=1}^{q} -p_j log_2 p_j - (1-p_j) log_2 (1-p_j).$$

$$\text{Donde: } p_j = \frac{\sum_{i=1}^{n} ||y_j \in Y_i||}{n}$$

Aqui p_j representa la fraccion de muestras de T con label y_j,

Ahora simplemente redefinimos la ganancia de informacion con la que armaremos el arbol de decision. La ganancia de inormacion dada por dividir T a lo largo del l-esimo feature separando por el valor θ es:

$$IG(\text{T}, l, \theta) = MLEnt(\text{T}) - \sum_{p \in -,+} \frac{|\text{T}^p|}{|\text{T}|} * MLEnt(\text{T}^p)$$

$$\text{T}^- = (x_i, Y_i \mid x_{il} \leq \theta, 1 \leq i \leq n)$$
$$\text{T}^+ = (x_i, Y_i \mid x_{il} > \theta, 1 \leq i \leq n)$$

La idea es, empezando por T = D, partir el dataset en T^- y T^+ segun los valores del l-esimo feature mayor (o menor) que θ. (Esto es, armar el nodo del arbol de decision). Esto se repite recursivamente con cada particion hasta que se cumpla algun criterio C.

Finalmente para una nueva instancia x, se la da como input al arbol de decision entrenado viajando por el hasta encontrar un nodo hoja relacionado con un numero de samples $\text{T} \subseteq D$. Luego, el conjunto de labels predicho corresponde a $Y = \{y_j \mid p_j > 0.5\ ,\ 1 \leq j \leq q\}$ Es decir, si en un nodo hoja la mayoria de los samples de entrenamiento que caen en ella son de tipo y_j, cualquier instancia de test que caiga en esa hoja tendra y_j como label relevante.

4.1 Conclusiones

En general pudimos observar como un cambio aparentemente menor con respecto a las hipotesis de lo problemas que veniamos viendo en la materia puede impactar significativamente, dando lugar a la necesidad de hallar o adaptar notablemente las metricas y algoritmos utilizados para poder resolver este nuevo problema. Esto nos da una idea de que tan amplio es el espectro de cosas que aún se pueden investigar.