Métodos de Aprendizaje Automático
Práctico 3

Bruno Garate & Nicolás Izquierdo & Guillermo Siriani

1. Clasificador Bayesiano Sencillo

En el Aprendizaje Bayesiano se infiere a partir de probabilidades. El objetivo es encontrar la hipótesis más probable dado un conjunto de datos de entrada. Para esto nos basamos en el teorema de Bayes:

Teorema de Bayes:

$$P(h|D) = \frac{P(D|h).P(h)}{P(D)}$$

Vamos a querer hallar el h que maximize $P(h|D)$. Ya que $P(D)$ es independiente de h solamente necesitamos considerar el término $P(D|h).P(h)$. El Clasificador Bayesiano Sencillo se basa en asumir que los valores de los atributos (condicionados a un atributo objetivo) son independientes entre sí, lo cual nos permitiría hallar la probabilidad condicionada de la observación como $P(a_1, a_2...a_n|v_j) = \prod_i P(a_i|v_j)$, siendo a_i el valor en la observación del atributo i-ésimo y $v_j \in V$ tal que V es el conjunto de todos los valores posibles del atributo objetivo.

Utilizando dicha igualdad, el Clasificador Bayesiano Sencillo se calcula como:

$$v_{NB} = argmax_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

1.1. Implementación

Fue implementada la clase Bayes (en el archivo bayes.py) que contiene el nombre del atributo objetivo, la lista de atributos considerados y los posibles valores que puede tomar cada uno de dichos atributos. Esta clase permite calcular todas las probabilidades necesarias para utilizar el Clasificador Bayesiano Sencillo y luego obtener la probabilidad de aciertos en un conjunto de prueba dadas las probabilidades calculadas.

1.2. Entrenamiento

El entrenamiento en este algoritmo consta del cálculo de un conjunto de probabilidades.

Por un lado se debe calcular la probabilidad de aparición de cada uno de los valores posibles del atributo objetivo. Para esto se utiliza el diccionario de valores posibles con los que se inicializa el objeto. Si para alguno de los valores posibles no se encuentran ejemplos en el conjunto de casos de entrenamiento que tomen dicho valor, se agrega la misma cantidad (por defecto 0, pero puede modificarse con el parámetro de entrada ajuste) de ejemplos para cada uno de los valores posibles del atributo objetivo, para evitar que haya valores con probabilidad 0.

Por otro lado se debe calcular la probabilidad de aparición de cada uno de los valores posibles de cada atributo, condicionadas a los distintos valores posibles del atributo objetivo. Para realizar estos cálculos se utilizan la lista de atributos y el diccionario de valores posibles con los que se inicializa el objeto. De forma análoga a las probabilidades calculadas anteriormente, se realizan ajustes a los valores en caso de que para algún valor posible de algún atributo no se encuentren ejemplos para alguno de los valores posibles del atributo objetivo.

El algoritmo de entrenamiento retorna una tupla con dos diccionarios de probabilidades, uno para la probabilidad de aparición de cada valor posible del atributo objetivo, y el otro con las probabilidades condicionadas de cada valor posible de cada atributo para cada valor posible del atributo objetivo.

1.3. Pruebas realizadas

Las pruebas se realizaron para 100 permutaciones distintas del conjunto de estudiantes de portugués y se calculó el promedio para cada una de las pruebas. En primer lugar se realizó una validación cruzada sobre 4/5 del conjunto de entrenamiento, tomando 9/10 de estos casos para entrenamiento y el 1/10 restante para validación. Luego se realizó un entrenamiento usando los 4/5 del conjunto original para entrenamiento y el 1/5 restante para validación. Esta última prueba se repitió para los valores de ajuste de 0 a 4. Como base para la evaluación del algoritmo se calculó además el porcentaje de acierto dando sistemáticamente la clase mas probable del conjunto de entrenamiento como resultado (Valor mas probable).

Evaluación	Porcentaje de aciertos
Valor mas probable	57.0775193798 %
Validación Cruzada	96.8588235294 %
Valor de ajuste 0 (default)	97.1627906977 %
Valor de ajuste 1	96.3875968992 %
Valor de ajuste 2	95.5503875969 %
Valor de ajuste 3	94.7364341085 %
Valor de ajuste 4	93.9612403101 %

Cuadro 1: Resultados de evaluación de Bayes

2. Aprendizaje basado en casos

El Aprendizaje basado en casos (Instanced-based learning), es una familia de algoritmos de aprendizaje y extensión de los algoritmos de clasificación k-NN.

Se destaca por clasificar nuevas instancias de un problema, utilizando otras instancias aprendidas en el entrenamiento. Se denomina aprendizaje basado en casos ya que no utiliza ningún tipo de modelo, sino que extrae todo el conocimiento necesario para resolver el problema de los casos almacenados en memoria en sí mismos.

2.1. K-Nearest Neighbors

Los algoritmos K-Nearest Neigbors se destacan por almacenar una cierta cantidad de instancias de un problema a modo de entrenamiento, para luego, en la etapa de evaluación, hallar los K casos mas cercanos a cada nueva instancia a evaluar.

Las ventajas de estos algoritmos radica en su velocidad de aprendizaje, fácil modificación de instancias de aprendizaje y la posibilidad de utilizar instancias de un problema tan complejas como se quiera sin perjudicar gravemente la eficiencia del algoritmo. Aunque sus ventajas son numerosas, los algoritmos k-NN pueden llegar a ser costosos en cuanto al tiempo de clasificación y ocupar excesiva memoria si el grupo de aprendizaje es demasiado grande, además la ocurrencia de atributos irrelevantes puede perjudicar la precisión de distancia entre instancias.

2.2. Implementación

Se implementó la clase knn (en el archivo Knn.py) la cual contiene todas las funciones necesarias para ejecutar el algoritmo, entre las cuales se destaca la función de distancia entre 2 valores de un mismo atributo, la función promedio, la cual evalúa el valor promedio entre los k neighbors, y la función de ponderación. El resto de las funciones se utilizan para la inicialización, la salida de datos y la modularización de los métodos.

En el caso de la función distancia, utilizamos la formula de distancia euclidiana para los atributos categorizables y la distancia de Hamming para los que se podrían representar como booleanos o que no eran categorizables. Luego para solucionar el problema de escala de los ejes, se decidió dividir cada distancia entre el rango de valores ocurridos en los ejemplos para así obtener un valor entre 0 y 1.

Para el cálculo del valor promedio se utilizó la función sugerida en el repartido teórico, en la cual se le otorga más relevancia a los vecinos más cercanos. Es decir, a menor distancia del vecino al objetivo, mayor repercución tendrá en el valor predicto.
La función está dada por :

$$h_x = \frac{\sum_{x_i \in Knn} W_i * f(x_i)}{\sum W_i}$$

Siendo W_i:

$$W_i = distancia(x_?, x_i)^{-2}$$

2.3. Entrenamiento

Con fines comparativos, se buscó utilizar una función ponderada para la función de evaluación. Se aplicó un algoritmo de gradiente de descenso estocástico para el aprendizaje de los pesos.

La función de error aplicada fue:

$$E = \begin{cases} \frac{h(x_1)-h(x_2)}{h(distancia(x_1,x_2)} & si\, distancia(x_1, x_2) \neq 0 \\ 100(h(x_1) - h(x_2)) & sino \end{cases}$$

2.4. Pruebas realizadas

Las pruebas se realizaron sobre el conjunto de datos de estudiantes de portugués (por ser el mayor de los dos) con atributo objetivo "G3". Para tener una buena estimación de la eficiencia del algoritmo se realizaron varias pruebas con diferentes combinaciones de datos de entrenamiento y de prueba.

Dado su efectividad en la anterior tarea, se vió importante aplicar a las pruebas validaciones cruzadas para aumentar aún más la veracidad de los resultados.

A pesar de haber realizado numerosas pruebas de eficiencia ajustando levemente el algoritmo y probando distintos tipos de ponderaciones, los resultados no fueron los esperados, ya que no se logró mejorar el porcentaje de aciertos obtenido retornando siempre el valor mas común. Tampoco fue posible hallar unas ponderaciones de los atributos que mejoraran el resultado considerablemente. En los mejores casos se logró una mejora en el porcentaje de aciertos del orden de 1 %.

Podríamos obtener dos conclusiones no necesariamente excluyentes a partir de estos datos. La primera es una incorrecta implementación del algoritmo o una selección de una función de error poco feliz. La otra posibilidad es que el resultado sea independiente de la métrica. Esto apuntaría a que KNN presenta dificultades al enfrentarse a estos casos de prueba.

3. Comparación de Resultados

Cada uno de los algoritmos se ejecutó para 100 permutaciones distintas de los estudiantes de portugués, y se calculó el promedio de aciertos para cada uno. En el caso del algoritmo ID3 se presentan los resultados de las validaciones sin establecer un límite de profundidad para los árboles construídos, pero se recuerda que los mejores resultados se obtenían al establecer que los árboles tuviesen solamente un nivel de profundidad, en cuyo caso el resultado coincidía con el de retornar siempre el valor más común.

Algoritmo	Validación cruzada	Validación Final
Valor mas común	56,8745098039 %	57,5503875969 %
ID3	43,3137254902 %	42,5813953488 %
Bayesiano	96,8588235294 %	97,1627906977 %
KNN-1	50,391176471 %	50,023255814 %
KNN-3	56,434313725 %	57,267442105 %

Cuadro 2: Resultados de evaluación de los distintos algoritmos

Se observa claramente que para estos experimentos los mejores resultados fueron obtenidos por el Clasificador Bayesiano Sencillo, que además es el que requiere menos tiempos de entrenamiento y ejecución. KNN-3 logra un resultado equivalente al de dar el valor mas común, lo que nos lleva a pensar que la distribución de los datos es tal que los casos más cercanos de cada ejemplo tienden a ser casos del valor mas común. Los otros algoritmos no obtuvieron resultados favorables en estos experimentos.

Para entender las discrepancias, es necesario considerar la plataforma diferente que presentan los diferentes algoritmos. Bayes presenta un clasificador generativo, mientras que KNN representa un clasificador discriminativo.

Mientras que KNN no toma asunciones sobre la distribución de los datos, salvo que son *localmente suaves*, bayes asume que las distribuciones son independientes. Mientras que bayes puede ajustarse perfectamente a una cantidad K de gausianas, KNN, dada una cantidad suficientemente grande de muestras, puede ajustarse a cualquier distribución arbitraria.

Por eso es siempre necesario entender que diferentes clasificadores responden a diferente distribuciones subyacentes , relaciones entre los datos, y características de los datos de muestra. La correcta selección de algoritmos dependerá entonces de los datos presentes así como el entorno de ejecución y los requisitos del mismo.

Referencias

[1] Liu Yang , *Distance Metric Learning: A Comprehensive Survey*,https://www.cs.cmu.edu/ liuy/frame_survey_v2.pdf

[2] Kilian Q. Weinberger ,Lawrence K. Saul, *Distance Metric Learning for Large Margin Nearest Neighbor Classification* ,http://jmlr.csail.mit.edu/papers/volume10/weinberger09a/weinberger09a.pdf

[3] *Difference between Probabilistic kNN and Naive Bayes* , http://stackoverflow.com/questions/36969740/difference-between-probabilistic-knn-and-naive-bayes