Aprendizaje Automático: Algoritmo k-NN

Enrique J. Carmona Suárez
ecarmona@dia.uned

Última versión: 07/10/2016

Dpto. Inteligencia Artificial, ETS de Ingeniería Informática, Universidad Nacional de Educación a Distancia (UNED), C/Juan del Rosal, 16, 28040-Madrid (Spain)

Índice

1. Introducción

Las técnicas pertenecientes al denominado paradigma de aprendizaje vago (del inglés *lazy learning*) son métodos de aprendizaje en los que no se infiere ningún modelo. En su lugar, el proceso de generalización a partir de los datos de entrenamiento se produce sólo en el momento en que se realiza una pregunta al sistema. De aquí, el nombre de aprendizaje *vago* o *perezoso* con el que se conoce a este tipo de algoritmos de aprendizaje.

Una de las principales ventajas del aprendizaje vago es que la respuesta a cada pregunta del sistema se obtiene mediante una aproximación local, es decir, a partir de los ejemplos de entrenamiento más parecidos, tal y como ocurre con el algoritmo k-NN. Esta propiedad de localidad se pierde en otros métodos de aprendizaje durante el proceso de generalización.

Por contra, una de las desventajas de este tipo de técnicas es que resultan normalmente lentas a la hora de responder a una pregunta del sistema, dado que, para ello, tienen que considerar todos y cada uno de los ejemplos del conjunto de entrenamiento. Sin embargo, no requieren de una fase de entrenamiento como la que sí utilizan todos aquellos paradigmas de aprendizaje basados en modelo. Otro inconveniente es que necesitan tener almacenados todos los ejemplos de entrenamiento, con el consiguiente requerimiento de espacio de almacenamiento en memoria.

2. Requisitos previos

1. Haber estudiado los capítulos 1 y 2 del texto base [Borrajo et al., 2006].

2. Haber estudiado la sección 6.3 (Capítulo 6), "Técnicas de aprendizaje vago", del texto base [Borrajo et al., 2006].

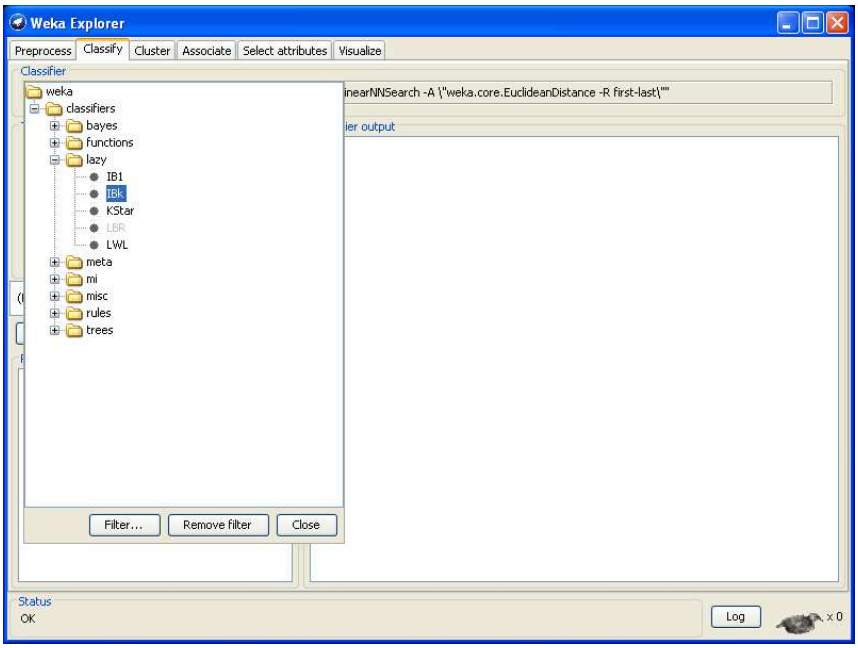

Figura 1: Acceso al algoritmo IBk desde el panel *Classify* de la ventana *Explorer*.

3. Objetivos

1. Familiarizarse con *Weka* en el uso de técnicas de aprendizaje vago.
2. Experimentar con el algoritmo IBk (versión Weka de k-NN).
3. Comprobar la dependencia del algoritmo k-NN respecto de sus parámetros de configuración:
   - Valor de k (número de vecinos más cercanos).
   - Ponderación por distancia.
4. Comprobar la dependencia del algoritmo k-NN respecto al número y poder de predicción de los atributos utilizados.

4. Material

- Programa Weka.
- Archivo de datos: "iris.arff" y "iris-unknown.arff".

5. Actividades

1. Cargar el fichero "iris.arff" desde el panel *Preprocess* de *Explorer*. El contenido de este archivo es una colección de instancias que representan 3 variedades de lirio. Se compone de cinco atributos. Los cuatros primeros corresponden a diversas medidas morfológicas de la planta (anchura y longitud de sépalos y pétalos) y el quinto corresponde al valor de la clase (tipo de lirio). Para más detalle, abra el archivo con cualquier editor de textos y consulte la información de cabecera de dicho archivo.
2. A continuación, seleccionaremos el algoritmo k-NN. Así, en la ventana *Explorer*, se seleccionará el panel *Classify* y, mediante el boton Choose, se elegirá el algoritmo *IBk* (version Weka de k-NN) ubicado dentro de la carpeta *lazy* (ver fig. 1).
3. Seguidamente, se describen los parámetros más relevantes del algoritmo IBk (ver fig. 2): (a) el parámetro *kNN* permite establecer el número de vecinos que usará el algoritmo; (b) el parámetro *crossValidate*, si se pone a *true* (por defecto está a *false*), permitirá al algoritmo

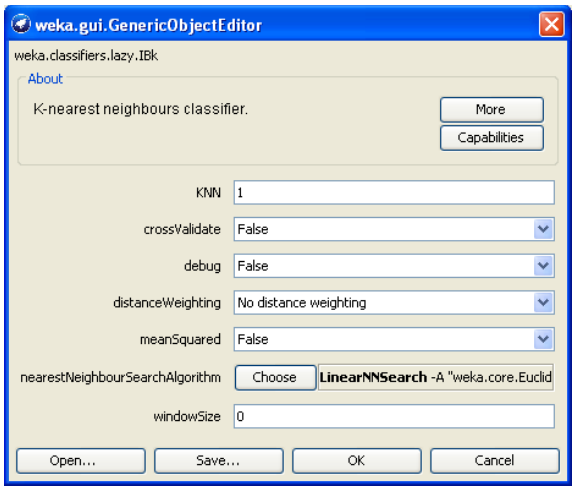

Figura 2: Parámetros del algoritmo IBk

seleccionar automáticamente el mejor valor para el número de vecinos a usar; (c) el parámetro *distanceWeighting* permite ponderar la contribución de cada vecino por su distancia (por defecto, no se aplica ponderación); (d) el parámetro *meanSquared* se dejará siempre a *false* (valor por defecto); (e) el parámetro *nearestNeighbourSearchAlgorithm* permite seleccionar el tipo de algoritmo de búsqueda del vecino más cercano y, en nuestro caso, siempre utilizaremos la opción por defecto: *LinearNNSearch* (algoritmo de búsqueda basado en fuerza bruta y que utiliza la distancia euclídea como medida de distancia); (f) finalmente, el parámetro *windowSize* se mantendrá siempre al valor 0 (valor por defecto). Se puede encontrar una descripción más detallada de estos parametros pulsando el botón More desde la ventana de parámetros (ver fig. 2).

4. Seleccione *kNN*=1 y el resto de parámetros a la opción por defecto. Realice los siguientes experimentos:

   - Experimento 4.1: Use la opción *Use training set* en la caja *Test options*. Ejecute el algoritmo.
   - Experimento 4.2: Seleccione ahora la opción *Percentaje split* y rellene el campo % con el valor 66. Esto hará que se utilice 2/3 de la base de datos para entrenamiento y el resto (1/3) para validación. En cada ejecución modifique el valor de la semilla utilizada para obtener diferentes particiones de entrenamiento/validación (este parámetro, *Random seed for XVal/ %Split*, está accesible pulsando el botón MoreOptions... situado en la caja *Test options*). Ejecute el análisis para los siguientes valores de la semilla: 1, 2, 3, 4 y 5.
   - Experimento 4.3: Finalmente, seleccione la opción *Cross-validation* y rellene el campo *Folds* a 10. Como en el apartado anterior, ejecute el algoritmo para los siguientes valores de semilla: 1, 2, 3, 4 y 5.
   - **Documentación a entregar:** Representar en una tabla el tanto por ciento de instancias correctamente clasificadas para cada experimento realizado y conteste a las siguientes preguntas: (1) Por qué el mayor porcentaje se obtiene en el primer experimento; (2) Por qué fluctúa dicho porcentaje en las distintas ejecuciones del segundo y tercer experimento; (3) Por qué fluctúa menos el mencionado porcentaje en el tercer experimento que en el segundo; (4) Por qué se dice que el resultado más fiable es el obtenido en el tercer experimento, el siguiente más fiable es el obtenido en el segundo y el menos fiable corresponde al obtenido en el primero; (5) Dado que el algoritmo k-NN no aprende ningún modelo, a qué se asocia el porcentaje de acierto de clasificación; (6) Muestre e interprete los resultados de la matriz de confusión obtenida en el experimento 4.3 (para semilla=1).

5. Dado que la medida de predicción de clasificación más robusta se obtiene con la técnica de validación cruzada, a partir de ahora, se trabajará siempre con la opción *Cross-validation* (*Folds*=10, semilla=1). En este apartado, se realizará un estudio de cómo influye el valor

del número de vecinos en el porcentaje de acierto de clasificación. Para ello, realice varias ejecuciones del algoritmo con los siguientes valores del parámetro *kNN*: 1,2,3,...10. El resto de parámetros permanecerán al valor por defecto.

- **Documentación a entregar:** Representar en una tabla el tanto por ciento de instancias correctamente clasificadas para cada experimento realizado y comente los resultados obtenidos. De otro lado, justifique si es cierta o no la siguiente afirmación: "*El algoritmo k-NN obtiene siempre mejores resultados de clasificación a medida que aumenta el valor de k*". En el caso de que considere que dicha afirmación no es cierta, utilice un contrajemplo para refutarla, es decir, invente y muestre una distribución de, por ejemplo, no más de 10 instancias (incluyendo instancias positivas y negativas) en el plano X-Y (2D). La idea es mostrar que si, en la distribución elegida de instancias positivas y negativas, cualquiera de ellas se considerara de clasificación desconocida, entonces dicha instancia sería clasificada correctamente con $k = 1$, pero erróneamente con $k = 3$.

6. Se puede también interpretar los resultados de clasificación desde un punto de vista gráfico. Así, dentro de la caja *Result List,* seleccione con el botón derecho del ratón la ejecución con la que, en el paso anterior, se produjo el mayor porcentaje correcto de clasificación y, sobre el menú desplegable, seleccione la opción *Visualize classifier errors* que, a su vez, abrirá una ventana gráfica. Esta ventana permite, entre otras cosas, comparar visualmente las clases reales de cada instancia con respecto a las clases predichas por el conjunto de datos de entrenamiento. Para ello, en las cajas correspondientes al eje X e Y, seleccione class(Nom) y *predictedclass(Nom)*, respectivamente (no olvide utilizar el botón deslizante denominado *Jitter* que permite introducir un pequeño ruido sobre el valor real de las instancias y, de esta manera, visualizar y separar las instancias solapadas). Otra funcionalidad de esta ventana es que permite conocer la información de la instancia asociada a cada uno de los puntos del plot sin más que clicar con el cursor del ratón sobre la instancia deseada. Para obtener información más detallada del resto de controles de esta ventana consultar el documento '*Weka: Guía de usuario de Explorer*'.

   - **Documentación a entregar**: represente y muestre la gráfica correspondiente a la clase real (eje X) respecto a la clase predicha (eje Y). Identifique e indique las instancias que están clasificadas erróneamente, proporcionando para cada una de ellas su etiqueta (número de orden en el conjunto de datos), su clase real y la clase predicha. Si lo considera relevante, incluya la representación de cualquier otra gráfica que considere oportuna, y comente el porqué de dicha relevancia.

7. En el paso 5, se tuvo que ejecutar el algoritmo tantas veces como valores diferentes se asignó al valor de k (parámetro kNN), con el objeto de decantarnos por el valor de k que mayor porcentaje correcto de clasificación obtuvo. No obstante, esto mismo se puede hacer de forma automática actuando sobre el parámetro *crossValidate* y asignándole el valor *true*. De esta forma, Weka realizará un barrido del algoritmo para todos los valores de k pertenecientes al conjunto $\{1, 2, ..., knn_0]$, siendo knn_0 el valor asignado al parámetro *kNN*. Por tanto, para este experimento se seleccionará $kNN = 10$, *crossValidate=true* y el resto de parámetros a su valor por defecto. A continuación, ejecute al algoritmo.

   - **Documentación a entregar**: Indicar el valor de k para el que se produce el mayor porcentaje de acierto de clasificación. ¿Coincide este valor de k con el del estudio realizado en el paso 5? ¿El porcentaje de acierto de clasificación obtenido con el k seleccionado automáticamente coincide con el porcentaje obtenido en el paso 5 cuando allí se utilizó el mismo valor de k? De no ser así, justifique el porqué de la discrepancia (Pista: utilice la información que proporciona Weka sobre el parámetro *crossValidate* al pulsar el botón More desde la ventana de parámetros).

8. Hasta ahora, todos los experimentos realizados no han usado la opción de ponderar la clasificación de cada vecino por su distancia. Esto es posible hacerlo actuando sobre el parámetro *distanceWeighting*. En este caso, seleccione $kNN = 10$, *crossValidate=true* y el resto de parámetros (salvo *distanceWeighting*) al valor por defecto. A continuación, realice los siguientes experimentos.

- Experimento 8.1: Seleccione *distanceWeighting="Weight by 1/distance"* y ejecute el algoritmo.
- Experimento 8.2: Seleccione *distanceWeighting="Weight by 1-distance"* y ejecute el algoritmo.
- **Documentación a entregar**: Indique el valor de k elegido en cada experimento y los porcentajes de acierto de clasificación respectivos. A la luz de los resultados obtenidos indique si, en este caso, se justifica la opción de ponderar por distancia (compare los resultados con los obtenidos en el paso 7).

9. Otra posibilidad es analizar el comportamiento del algoritmo en relación al número y poder de predicción del conjunto de atributos usado. Para ello, se puede realizar, por ejemplo, distintos experimentos en los que, en cada uno de ellos, se seleccionará una pareja de atributos de los 4 posibles (sin contar el atributo clase). Para ello, seleccione $kNN = 10$, *crossValidate=true* y el resto de parámetros al valor por defecto. A continuación, realice los siguientes experimentos.

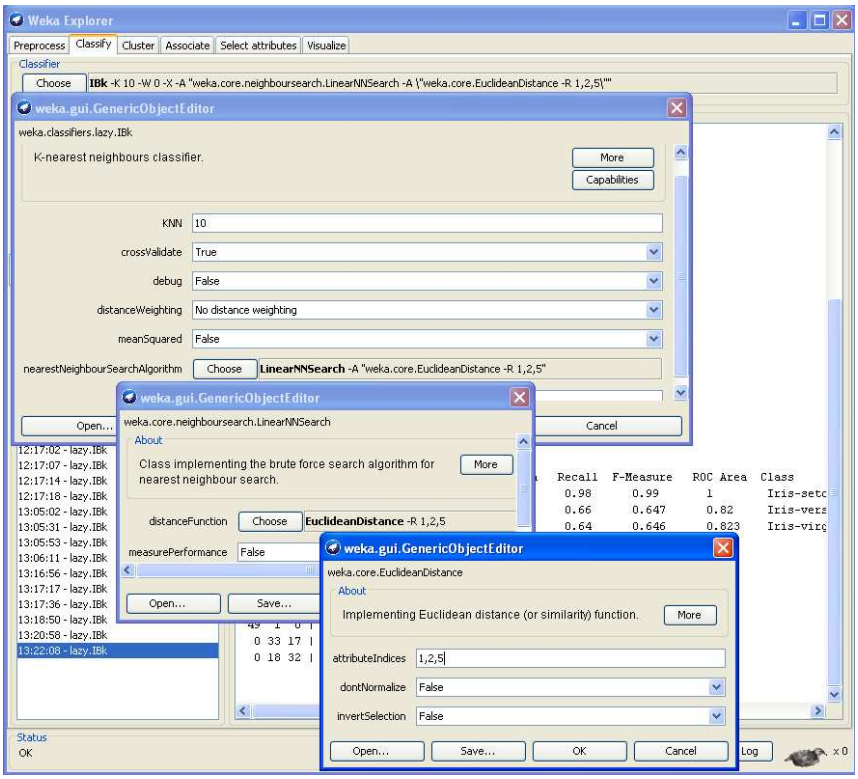

Figura 3: Ruta de subventanas para acceder al parámetro *atributteIndices* de la función *Euclidean distance*.

- Experimento 9.1. Teniendo en cuenta que los atributos tienen la siguiente asignación numérica: 1.sepallength, 2.sepalwidth, 3.petallength, 4.petalwidth, 5.class, seleccione sólo los atributos 1,2,5 y ejecute el algoritmo. Aunque el filtrado de los atributos 3 y 4 podría realizarse desde el panel *Preprocess* de Weka, en este caso, hay una forma más rápida de hacerlo. Para ello, basta especificar en el parámetro *nearestNeighbourSearchAlgorithm* > *LinearNNSearch* > *EuclideanDistance* > *atributteIndices* los índices de los atributos a utilizar (separados por comas) en el cálculo de la distancia (ver fig. 3).
- Experimento 9.2. Repetir el experimento 9.1, seleccionando sólo los atributos 1,3,5.
- Experimento 9.3. Repetir el experimento 9.1, seleccionando sólo los atributos 1,4,5.
- Experimento 9.4. Repetir el experimento 9.1, seleccionando sólo los atributos 2,3,5.
- Experimento 9.5. Repetir el experimento 9.1, seleccionando sólo los atributos 2,4,5.
- Experimento 9.2. Repetir el experimento 9.1, seleccionando sólo los atributos 3,4,5.

- **Documentación a entregar**: Indique el valor de k elegido en cada experimento y los porcentajes de acierto de clasificación respectivos. A la luz de los resultados obtenidos, indique si el algoritmo k-NN depende o no del subconjunto de atributos (elegido a partir del conjunto de atributos original). Si la respuesta es afirmativa, ¿existe alguna pareja de atributos que permita obtener un porcentaje correcto de clasificación tan o más competitivo como el que se obtuvo trabajando con todos los atributos del conjunto de datos? Compare los resultados con los obtenidos en el paso 7.

10. No hay que olvidar que en los métodos pertenecientes a la técnica de aprendizaje vago es el propio conjunto de datos de entrenamiento el que almacena el conocimiento a usar. En este apartado se va a utilizar dicho conocimiento para clasificar un conjunto de instancias de clasificación desconocida y almacenadas en el fichero "iris-unknown.arff". Si se edita este archivo[1], puede comprobarse que, en la sección de datos, el valor del atributo clase está etiquetado con la marca '?' (valor desconocido). A continuación, realice las siguiente acciones:

    - Asigne al parámetro *kNN* el mejor valor obtenido en todos los experimentos, ponga *crossValidate=false,* active o no el parámetro *distanceWeighting* dependiendo de si el efecto de la ponderación fue o no beneficioso, seleccione los atributos a usar y el resto de parámetros manténgalos a su valor por defecto.
    - Ejecute el algoritmo con los valores de parámetros elegidos.
    - Marque la opción *Supplied test set* de la caja *Test options* y, a continuación, pulse el botón Set... Esto le permitirá cargar en memoria el fichero de instancias de clasificación desconocida (iris-unknown.arff) y dejarlo preparado para la acción siguiente.
    - Sitúe el cursor encima de la ejecución realizada (caja *Result List*) y, con el botón derecho del ratón, seleccione la opción *Re-evaluate model on current test set.* Esto producirá la clasificación de las instancias del mencionado fichero, pero no se generará ningún tipo de estadística.
    - Para saber cómo se clasificaron las nuevas instancias, se seleccionará la opción *Visualize Classifier Error* asociada a la ejecución realizada (tal y como ya se indicó en el paso 6). Esto provocará la aparición de una nueva ventana que muestra un plot con todas las instancias implicadas[2]. Utilice el botón Save de dicha ventana para guardar la información de clasificación en un archivo. Ahora, si se edita el archivo generado, podrá comprobarse que el contenido coincide con el archivo original con la salvedad de que se ha añadido un nuevo atributo, denominado *predictedClass,* que muestra la clasificación predicha para cada instancia.
    - **Documentación a entregar**: muestre el contenido del archivo generado.

6. Conclusiones

En la documentación a entregar, enumere finalmente de forma esquemática las conclusiones derivadas de los resultados prácticos obtenidos con la realización de la prácticas. Hágalo tanto desde un punto de vista general, relativo a la tarea de clasificación mediante la técnica de aprendizaje vago, como desde un punto de vista particular, referido al algoritmo k-NN.

Referencias

[Borrajo et al., 2006] Borrajo, D., González, J., & Isasi, P. (2006). *Aprendizaje Automático.* Madrid (España): Sanz y Torres.

[1] Se puede editar con cualquier editor de textos

[2] Si se hace clic con el botón izquierdo del ratón sobre cualquiera de las instancias representadas en el plot, se puede chequear la predicción de clasificación (ver atributo *predictedClass*) realizada por el modelo para la instancia seleccionada.