uc3m | Universidad Carlos III de Madrid

OPENCOURSEWARE

APRENDIZAJE AUTOMÁTICO PARA EL ANÁLISIS DE DATOS

GRADO EN ESTADÍSTICA Y EMPRESA

Ricardo Aler

1. El método de Sequential Model-based Optimization (SMO) construye un modelo de regresión a partir de puntos explorados previamente. ¿Para qué usa SMO ese modelo y cómo?

SMO usa ese modelo *f* para proponer nuevos puntos *xi* a explorar (donde los puntos son combinaciones de hiper-parámetros). Si el objetivo es minimizar un error (RMSE por ejemplo), los nuevos puntos propuestos serán los mínimos *xi* de ese modelo *f*.

2. Si f es una neurona de una red de neuronas y tiene 3 entradas x1, x2, x3, ¿cuales serían las operaciones matemáticas f que realiza la neurona para computar la salida y? (y=f(x1, x2, x3)).

y = sigmoide(w1*x1+w2*x2+w3*x3+w4)
La salida de una neurona es la sigmoide de la suma de las entradas ponderada por los pesos wi (mas el bias w4)

3. ¿Boosting puede tener problemas de sobreaprendizaje al ruido en problemas de clasificación? Explicar la respuesta.

Si. La razón es que en Boosting, el siguiente modelo del ensemble da más peso a los datos mal clasificados por el anterior modelo. Los datos con ruido son, en el fondo, imposibles de clasificar correctamente, por lo que irán teniendo pesos grandes y los sucesivos modelos del ensemble se centrarán en ellos, pudiendo acabar memorizando (overfitting) esos datos con ruido.

4. Supongamos que tenemos un conjunto de datos disponibles D y queremos construir y evaluar un modelo usando sólo los atributos más relevantes. Supongamos que seguimos el siguiente proceso: 1) usamos un método de selección de atributos usando el conjunto D y elegimos los mejores; 2) después dividimos D en una partición de entrenamiento y otra de test; 3) construimos el modelo con la de entrenamiento y lo evaluamos con la de test. ¿Es correcto este proceso? ¿Por qué?

El procedimiento es incorrecto porque en el primer paso (selección de atributos) se están usando datos que en el paso 3 estarán en la partición de test. Es decir, para saber qué atributos son relevantes como entradas del modelo, se está utilizando información de los datos de test y por tanto la partición de test ya no sería independiente del proceso de construcción del modelo.

5. ¿En qué se parecen y en qué se diferencian los Extremely-randomized trees de los Random Forests?

El RF estándar usa dos métodos de aleatorización para construir los distintos modelos del ensemble:
— Creación de varias conjuntos mediante muestreo con reemplazo
— Selección aleatoria de m atributos de entre los M, antes de elegir el mejor

Los extra-trees funcionan de manera similar, excepto que:
— Se usa el conjunto de entrenamiento original para todos los miembros del ensemble en lugar de crear varias conjuntos mediante muestreo.
— Al igual que con Random Forests, se eligen m atributos aleatoriamente, pero **la mayor diferencia** es que si el atributo es numérico, el punto de corte (threshold) se elige aleatoriamente.

6. Supongamos que tenemos unos datos disponibles D para un problema de clasificación bi-clase, y queremos construir dos particiones ESTRATIFICADAS: entrenamiento (⅔) y test (⅓). Describir el procedimiento que habría que utilizar para construir esas particiones.

El objetivo de las particiones estratificadas es que su distribución de clases sea similar a la presente en los datos originales y es conveniente usarlas para problemas con muestra desbalanceada. Para conseguirlas podemos seguir el siguiente procedimiento: obtenemos el conjunto de los datos positivos P y el conjunto de los datos negativos N. Seleccionamos ⅔ de los datos de P y ⅔ de los datos de N para la partición de entrenamiento. Seleccionamos ⅓ de los datos de P y ⅓ de los datos N para la partición de test.

7. El proceso de entrenamiento de una SVM lineal involucra minimizar esta expresión: (1/margen^2) + C*Σξi . ¿Qué le pasará a una SVM si hacemos que el hiper-parámetro C sea muy grande? ¿Por qué?

C multiplica a la suma de las variables de holgura, las cuales representan los errores de los datos mal clasificados por el modelo. Si C es muy grande, estamos forzando a que el modelo cometa muy pocos errores. Esto fuerza al modelo a aprenderse datos ruidosos y reducir la capacidad de generalización del mismo (es decir, sobreaprendizaje).

Otra manera de verlo es que si C es grande, se va a minimizar preferentemente el objetivo Σξi frente al objetivo 1/margen, lo que resultará en un margen de separación entre las clases demasiado pequeño y con mala generalización.

8. Supongamos que queremos utilizar Map/Reduce para contar cuantas veces aparece cada palabra en un fichero que contiene todos los textos de la literatura universal. Explicar con palabras el proceso que se seguiria para contar palabras y en concreto, qué es lo que tendría que hacer la función Map, qué la función Reduce y qué sort-and-shuffle.

Map se ejecutaría en cada ordenador local. Map descompondría cada línea de los ficheros de texto en palabras y para cada palabra, enviaría (clave=palabra, valor=1) a la red. Sort-and-suffle agruparía las parejas (clave,valor) por clave (o sea, por palabra) y enviaría cada grupo a un reduce distinto. Cada reduce recibiría los valores correspondientes a cada palabra y los sumaría (o sea, sumaría los unos correspondientes a esa palabra, lo que resulta en el número de veces que aparece esa palabra).

9. En el algoritmo del vecino más cercano, para qué valores de K (número de vecinos) se puede producir más sobre-aprendizaje? ¿para valores pequeños o para valores grandes? ¿Por qué?

Para valores pequeños se puede producir más sobre-aprendizaje. Para k=1, la clasificación depende de un único dato, el más cercano. Si ese dato es ruidoso, la clasificación va a ser incorrecta, por lo que se está produciendo sobre-adaptación al ruido. Aumentando el valor del número de vecinos (k) se disminuye la influencia de los datos ruidosos y por tanto se disminuye el sobre-aprendizaje.

Otra manera de verlo es que cuanto más complejo es el modelo (o más compleja la frontera de separación), más fácil es que haya sobre-ajuste. Y en el caso de KNN, cuanto mayor es K, más suave es la frontera.

10. Supongamos que un clasificador trivial clasifica las instancias como positivas con probabilidad 0.9 y como negativas con probabilidad 0.1. ¿Cuál sería el True Negative Rate de este clasificador? ¿Por qué?

Se puede ver que este clasificador clasificará como negativas el 0.1 de las instancias negativas, por tanto su TNR = 0.1.

El TNR es la proporción de instancias negativas clasificadas (correctamente) como negativas. De x instancias negativas, 0.9*x serán clasificadas (incorrectamente) como positivas y 0.1*x como negativas (correctamente). Por tanto, la tasa de aciertos para los negativos TNR es 0.1*x/x = 0.1.