UNIDAD TEMÁTICA 3: Algoritmos No Lineales

Trabajo de Aplicación 6 – Support Vector Machines

Ejercicio 1

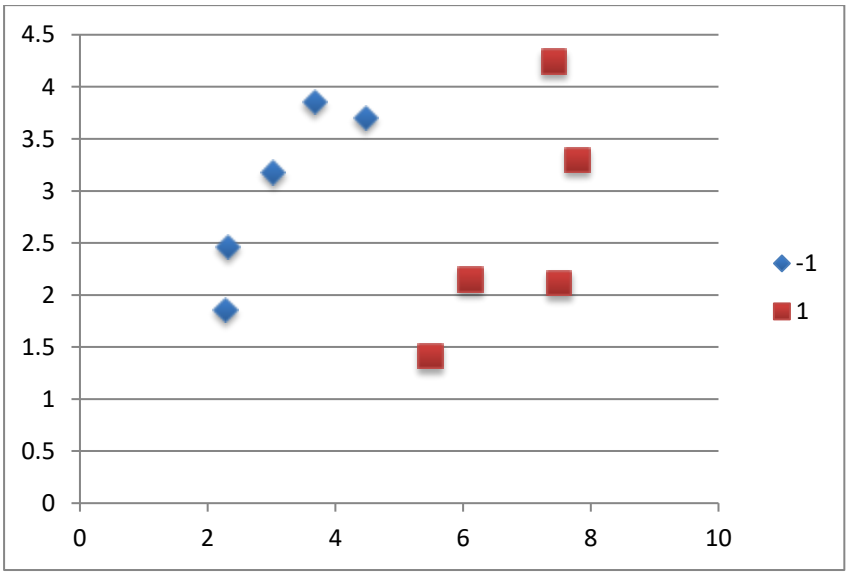
En este ejercicio practicaremos el algoritmo de Support Vector Machines paso a paso, utilizando descenso de sub-gradiente. O sea que utilizaremos el descenso de sub-gradiente para

Actualizar los coeficientes del modelo SVM

Iterativamente, aprender el modelo SVM a partir de los datos de entrenamiento

Y luego veremos cómo utilizar el modelo SVM generado para hacer predicciones.

1. utilizando la planilla electrónica adjunta (TA6-EJ1.xlsx),
   - graficar los datos de entrenamiento como puntos, con dos series,
     - para Y=-1 e Y = 1
   - ¿cómo se ven los ejemplos en cuanto a su clasificación?



Vemos que los ejemplos están bien separados en las clases (el ejemplo es artificial)

Forma del modelo SVM Lineal

El modelo SVM lineal es una recta, y el objetivo del algoritmo de aprendizaje es encontrar valores de los coeficientes de esta recta tal que separen de la mejor forma las clases.

La recta puede expresarse como

- B0 + B1 × **X1** + B2 × **X2** **= 0**
- Donde B0, B1 y B2 son los coeficientes a hallar, y X1, X2 las variables de entrada.
- Para simplificar, podemos hacer B0 = 0 (descartar el término independiente o bias) con lo que la recta ha de pasar por el origen.

Método de optimización de SVM

Utilizaremos en este ejercicio el método del descenso de sub-gradiente, para aprender los coeficientes de un SVM lineal. Este método utiliza un patrón seleccionado aleatoriamente en cada iteración, y utilizado para actualizar los coeficientes. Después de un número de iteraciones grande (miles o cientos de miles) , el algoritmo se estabilizará sobre un cierto conjunto de coeficientes.

En primer lugar se calcula un valor de salida, como

- **Salida** = **Y** x (B1 × **X1** + B2 × **X2**)

Se utilizan dos procedimientos de actualización diferentes, dependiendo del valor de salida. Si la salida es mayor que 1, entonces indica que el patrón de entrenamiento no es un vector de soporte. Esto significa que la instancia no estaba directamente involucrada en el cálculo de la salida, en cuyo caso se decrementan levemente los pesos:

$$b = (1 - \frac{1}{t}) \times b$$

En donde ***b*** es el coeficiente que se está actualizando (B1 or B2), y ***t*** es la iteración actual. Si la salida es menor a 1 entonces se assume que la instancia de entrenamiento es un vector de soporte y debe ser actualizdo para explicar mejor los datos.

$$b = (1 - \frac{1}{t}) \times b + \frac{1}{lambda \times t} \times (y \times x)$$

En donde ***b*** es el coeficiente que se está actualizando, ***t*** es la iteración actual y ***lambda*** es un parámetro del algoritmo de aprendizaje, habitualmente con valores pequeños (0.0001 o menor).

El procedimiento se repite hasta que la tasa de error cae a un valor aceptable o hasta que se alcanza un cierto número máximo de iteraciones. Las tasas de aprendizaje menores a menudo requieren muchas más iteraciones. La gran cantidad de iteraciones que requiere este algoritmo es una de sus desventajas.

Aprendizaje del modelo SVM a partir de los datos de entrenamiento.

Realizaremos unas cuantas iteraciones para ilustrar el funcionamiento del algoritmo, usando un lambda muy grande: 0.45 (este valor es muy grande y provocará grandes cambios en cada iteración; normalmente los valores de lambda se toman muy pequeños).

Iteración de aprendizaje #1

Se comienza inicializando los coeficientes en 0, B1 = 0.0 y B2 = 0.0. Inicializar también el contador de iteraciones **t=1.**

Idealmente se tomarían los ejemplos en forma aleatoria del dataset, para evitar que el algoritmo se bloquee. En este ejercicio los tomaremos en el orden en que aparecen, para mayor facilidad de visualización.

Con el primer patrón, calculamos el valor de salida para la iteración:

- **Salida** = **Y** x (B1 × **X1** + B2 × **X2**)
- **Salida** = **-1** x (0.0 × **2.327868056** + 0.0 × **2.458016525**)
- **Salida** = 0.0

La salida es menor que 1, así que utilizamos el procedimiento más complejo para actualizar:

$$b = (1 - \frac{1}{t}) \times b + \frac{1}{lambda \times t} \times (y \times x)$$

B1 = (1 – 1/1) × **B1** + 1/(0.45 x 1) x (-1 × **2.327868056**)

B1 **=** -5.173040124

Y B2 = -5.462258944

Iteración de aprendizaje #2

Ahora t=2, y utilizaremos los valores calculados de los coeficientes en la iteracion anterior, usando el Segundo ejemplo del dataset:

- Salida = -1 x (-5.173040124× 3.032830419+ -5.462258944× 3.170770366)
- Salida = 33.00852224

La salida es mayor que 1, lo que sugiere que este ejemplo no es un vector de soporte. Aplicamos el primer procedimiento entonces:

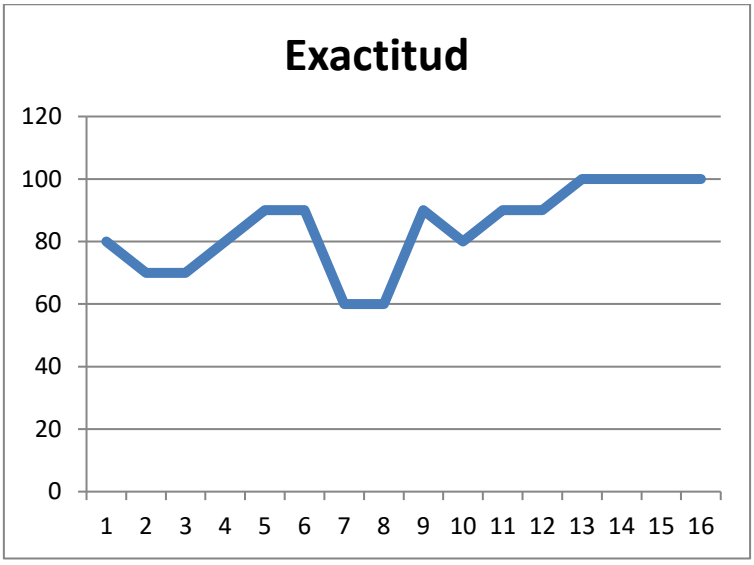
$$b = (1 - \frac{1}{t}) \times b$$

B1 = (1 – 1/2) × -5.173040124 = **-2.586520062**

B2 = **-2.731129472**

Más iteraciones

- Repite el proceso para el resto del dataset. Cada pasada sobre el dataset es llamada “época”.
- Repite todo el proceso por al menos unas 15 épocas más. En cada época, calcula la exactitud de entrenamiento.
- Realiza un gráfico que vaya mostrando cómo varía la exactitude de entrenamiento en función de las épocas.
- ¿En qué momento se alcanza una exactitude de 100%?



Al finalizar todas las épocas, tenemos los valores finales de los coeficientes del modelo:

- **B1 = 0.552391765**
- **B2 = -0.724533592**

LA FORMA DEL HIPERPLANO APRENDIDO ES, ENTONCES,

$$B0 + B1 \times \mathbf{X1} + B2 \times \mathbf{X2} = \mathbf{0}$$

$$0 + (\mathbf{0.552391765\ x\ X1}) + (\mathbf{-0.724533592\ x\ X2})$$

Utilizar el modelo aprendido para hacer predicciones y calcular su exactitud

Ahora que hemos obtenido los coeficientes para la recta, podemos utilizar el modelo para predicción.

Lo vamos a probar con el mismo conjunto de entrenamiento, pero fácilmente se puede adaptar para hacer predicciones sobre datos nuevos (se sugiere realizar esto como ejercicio domiciliario).

Las predicciones se realizarán utilizando las siguientes ecuaciones:

- **salida** = B1 × **X1** + B2 × **X2**
- **Si salida < 0, Y = -1**
- **Si salida > 0, Y = +1**

- Utiliza los datos de entrenamiento para calcular las predicciones.
- Para cada ejemplo, calcula el error como la diferencia entre **Y** (dato) y la predicción obtenida.
- Al finalizar con todos los ejemplos, calcula la **exactitud** de la predicción.