# OCR-Weka: Support Vector Machines - SVM

**NOMBRE: Frédéric Roux** **Fecha:13/12/2018**

En la carpeta *classifiers/functions* de la pestaña *Classify* de Weka puedes encontrar el clasificador *SMO*, la implementación del modelo Support Vector Machine en Weka.

(a) Los parámetros principales del clasificador SMO son la complejidad del modelo (parámetro *c*) y el tipo de kernel utilizado. Teniendo en cuenta lo visto en teoría, explica dichos parámetros y explica los distintos tipos de kernels que Weka tiene implementados.

**Complejidad del modelo (parámetro *c*) :** SVM pretende encontrar la solución que maximice la distancia entre el hiperplano y las clases. Los SVMs asumen separabilidad lineal entre dos clases (hard margin). Pero tambien Se pueden afrontar problemas linealmente no separables (mediante soft margin o una transformación, kerneltrick). Con el parámetro c se relaja la restricción y se permiten puntos mal clasificados pero penalizando la función objetivo. La formulación primaria sería:

**función objetivo**

$$\frac{1}{2}\sum_{i=1}^{n} w_i^2 + C\sum_{i=1}^{N} \xi_i$$

**restricciones**

$$y_i(\vec{w}\cdot\vec{x_i}+b) \geqslant 1-\xi_i \quad \text{con} \quad i=1,\ldots,N$$

**Kernel trick:** Un kernel K es un producto interno en un espacio de características.

$$K(\vec{x_i},\vec{x_j}) = \Phi(\vec{x_i})\cdot\Phi(\vec{x_j})$$

Cuando los datos no son linealmente separables se transforman los datos a un espacio con más dimensiones mediante un kernel de manera que las clases sean linealmente separables en el nuevo espacio. Se busca el hiperplano de separación óptimo en ese nuevo espacio de representación. Luego se proyecta sobre el espacio de clasificación original.

**Kernels en Weka:**

***NormalizedPolyKernel:** K(x,y) = <x,y>/sqrt(<x,x><y,y>) where <x,y> = PolyKernel(x,y)
Es el mismo que el PolyKernel, pero normalizado por sqrt(<x,x><y,y>).

***PolyKernel:** K(x, y) = <x, y>^p or K(x, y) = (<x, y>+1)^p
Este kernel, y el kernel RBF, son los kernels mas usado en el contexto de SVMs y permiten de transformar datos que no son separables linealmente a un espacio en cual si se pueden

separar linealmente.

***PrecomputedMatrixKernel:** matriz con kernel custom que se puede cargare desde un archivo

***PUK:** implementacion de la funcion Pearson VII que pertenece al sistema de funciones Pearson que permiten de hacer un fit a los 4 parametros claves de distribuciones (media, varianza, asimetria y curtosis).

$$I(2\theta) = I_{\max} \frac{w^{2m}}{[\, w^2 + (2^{1/m} - 1)\,(2\theta - 2\theta_0)^2 \,]^{\,m}}$$

***RBFkernel:** K(x, y) = exp(-gamma*(x-y)^2)
Este es el kernel mas usado con SVMs. La diferencia x-y mide la similitud entre dos casos xi e yi en el espacio 2-D definido por x e y. Es una manera de proyectar los casos a un espacio en cual están re-organizados geográficamente por su similitud.

***StringKernel:** implementacion de una funcion que permite de medir la similitud de datos de tipo texto (string)

(b) Prueba 6 diferentes opciones de dichos parámetros e indica la tasa de aciertos obtenida para el conjunto de test. Para ello, rellena la siguiente tabla de resultados. Recuerda, como siempre, entrenar los clasificadores con el conjunto de entrenamiento.

| **Parámetros** | **Tasa de acierto (%)** |
|---|---|
| RBFkernel, c = 0.9 | 70% |
| RBFkernel, c = 1 | 69% |
| RBFkernel, c = 1.1 | 74% |
| Polykernel, c = 0.9 | 81% |
| Polykernel, c = 1 | 81% |
| Polykernel, c = 1.1 | 81% |

(c) Haz algún comentario acerca de la matriz de confusión obtenida en alguno de los casos. ¿Cuál es la clase más "confusa"?

**RBFkernel, c = 1.1**

=== Confusion Matrix ===

```
 a  b  c  d  e  f  g  h  i  j   <-- classified as
10  0  0  0  0  0  0  0  0  0 |  a = 0
 0 10  0  0  0  0  0  0  0  0 |  b = 1
 1  1  4  0  0  0  4  0  0  0 |  c = 2
 1  0  0  9  0  0  0  0  0  0 |  d = 3
 0  0  0  0  6  0  0  0  0  4 |  e = 4
 0  0  0  0  0 10  0  0  0  0 |  f = 5
 0  0  0  0  0  1  9  0  0  0 |  g = 6
 0  0  0  0  1  0  0  6  0  3 |  h = 7
 0  0  1  0  1  3  0  0  5  0 |  i = 8
 0  0  0  0  3  0  0  2  0  5 |  j = 9
```

**[clase mas confusa] TPR: 0.4**

**Polykernel, c = 1.1**

=== Confusion Matrix ===

```
 a  b  c  d  e  f  g  h  i  j   <-- classified as
10  0  0  0  0  0  0  0  0  0 |  a = 0
 0 10  0  0  0  0  0  0  0  0 |  b = 1
 1  2  4  0  0  0  2  0  1  0 |  c = 2
 0  0  0 10  0  0  0  0  0  0 |  d = 3
 0  0  0  0 10  0  0  0  0  0 |  e = 4
 0  0  0  0  0  8  2  0  0  0 |  f = 5
 0  0  0  0  0  0 10  0  0  0 |  g = 6
 0  0  0  0  0  0  0  8  1  1 |  h = 7
 0  0  1  0  1  0  0  0  8  0 |  i = 8
 0  0  0  0  7  0  0  0  0  3 |  j = 9
```

**[clase mas confusa] TPR: 0.4**

Las matrices de confusión indican que para la mayoría de las clases se obtiene una tasa de acierto muy alta (>70%). La única clase para cual se obtiene una tasa de acierto muy baja (<50%) es la clase que

corresponde al numero 2.