Clasificador SVM productivo

1. Vamos a usar LIBSVM para entrenar el modelo.

   Para ello descomprimimos `/miscelaneos/libsvm-3.22.zip` y accedemos al directorio `/libsvm-3.22/`.

   a) Escalamos los datos del conjunto de entrenamiento:

   ```
   ./svm-scale -s scal_params empty.tr.800.libsvm >train.scale
   ```

   `empty.tr.800.libsvm` es un subconjunto del dataset `empty.all.csv` (usado en la notebook `clase-svm-complemento-slides`) en formato libsvm.

   b) Escalamos los datos del conjunto de test en función de los valores obtenidos para en conjunto de entrenamiento:

   ```
   ./svm-scale -r scal_params empty.val.libsvm >test.scale
   ```

   `empty.val.libsvm` es un subconjunto del dataset `empty.all.csv` (usado en la notebook `clase-svm-complemento-slides`) en formato libsvm.

   c) Entrenamos un clasificador (`-s 0`) lineal (`-t 0`) con valor de $C = 32$, sobre el conjunto de datos escalado:

   ```
   ./svm-train -s 0 -t 0 -c 32 train.scale
   ...*.*
   optimization finished, #iter = 4728
   nu = 0.010409
   obj = -309.199181, rho = 5.483705
   nSV = 50, nBSV = 2
   Total nSV = 50
   ```

   d) Testeamos el clasificador sobre el conjunto de datos de test escalado:

   ```
   ./svm-predict test.scale train.scale.model test.output
   Accuracy = 99.5% (199/200) (classification)
   ```

   En `test.output` deja en cada línea, un valor de `1` o `-1` que corresponde a la predicción del clasificador para el elemento que estaba en esa línea.

2. Con el modelo almacenado en `train.scale.model` podemos ahora independizarnos de LIBSVM implementando con los datos allí disponibles la frontera de decisión $\mathbf{w}^T\mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$ que se describe en la transparencia 28 de la teoría.

3. Una posible implementación en GNU Awk es `/miscelaneos/svm_productivo.awk`.

   ```
   awk -f svm_productivo.awk test.scale >test.awk.output
   ```

4. `vimdiff test.output test.awk.output`