UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Práctica 2: SVMs

Alfons Juan, Jorge Civera

DSIC

Departament de Sistemes Informàtics i Computació

Índice

1. Introducción

- En esta práctica aplicamos SVMs (a MNIST) con el paquete ***svm*** de la librería ***sklearn***, basado en la librería ***libsvm***, cuya guía se recomienda para más detalles de los que se dan seguidamente
- Primero veremos una breve revisión teórica para contextualizar los aspectos más relevantes en el entrenamiento de SVMs
- Segundo veremos algunas actividades guiadas para el entrenamiento de SVMs con kernel lineal en MNIST
- Por último se propone un ejercicio a realizar que, en esencia, consiste en repetir con otros kernels las actividades hechas
- Se dispone de una tarea PoliformaT para entregar el resultado del ejercicio en el plazo de la práctica

2. Breve revisión teórica

- Dadas N muestras $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, $\boldsymbol{x}_n \in \mathbb{R}^D$, $\boldsymbol{y} \in \{-1,+1\}^N$, las SVMs usuales requieren resolver el problema de optimización:

$$\begin{bmatrix}\boldsymbol{w}^*\\ w_0^*\\ \boldsymbol{\xi}^*\end{bmatrix} = \underset{\boldsymbol{w},w_0,\boldsymbol{\xi}}{\arg\min}\ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_n \xi_n \quad \text{s.a.}\ \{y_n f_n \geq 1-\xi_n,\ \xi_n \geq 0\}$$

  donde

  - $f_n \triangleq \boldsymbol{w}^t\phi(\boldsymbol{x}_n) + w_0$ es el resultado de clasificar linealmente una transformación ϕ de $\boldsymbol{x}_n$ a algún espacio (de alta dimensión)
  - $C \geq 0$ es el parámetro de ***penalización*** o ***regularización***
    - $\mapsto$ $C \uparrow$: $\boldsymbol{\xi}^* = \mathbf{0}$, restricciones fuertes, menor regularización
    - $\mapsto$ $C \downarrow$: $\boldsymbol{\xi}^* \geq \mathbf{0}$, restricciones débiles, mayor regularización

- La resolución del problema requiere calcular una función ***kernel***, $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_j)$, con alguno de cuatro modelos básicos:

  - ***Lineal:*** $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^t \boldsymbol{x}_j$

  - ***Gaussiano (o RBF):*** $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2), \;\; \gamma > 0$
    - $\mapsto$ Cumple $0 < \mathcal{K}_{ij} \leq 1$
    - $\mapsto$ $\gamma \uparrow$: kernel estrecho, requiere mayor regularización
    - $\mapsto$ $\gamma \downarrow$: kernel ancho, requiere menor regularización

  - ***Polinómico:*** $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma \boldsymbol{x}_i^t \boldsymbol{x}_j + r)^d, \;\; \gamma > 0$
    - $\mapsto$ $d \uparrow$: tiende a ∞ si $\gamma \boldsymbol{x}_i^t \boldsymbol{x}_j + r > 1$; a 0 si $\gamma \boldsymbol{x}_i^t \boldsymbol{x}_j + r < 1$

  - ***Sigmoide:*** $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\gamma \boldsymbol{x}_i^t \boldsymbol{x}_j + r)$

- La aplicación de SVMs a una tarea requiere elegir un kernel apropiado, ajustar C y, si procede, también γ, d (degree) y r (coef0)

3. Actividades

3.1. Tarea MNIST

- Vamos a aplicar el clasificador de mixturas de Gaussianas a MNIST proyectada mediante PCA a 20 dimensiones

- Descarga los siguientes ficheros de PoliformaT:
  - `train-images-idx3-ubyte.pca20.npz` (vectores)
  - `train-labels-idx1-ubyte.pca20.npz` (etiquetas)

3.2. Training y dev

- Script para obtener una partición tr-dev a partir de MNIST:

part.py

```
#!/usr/bin/env python
import sys
import numpy as np

if len(sys.argv)!=8:
  print('%s <dat> <lab> <tr%%> <dv%%> <seed> <tr> <dv>' % sys.argv[0]);
  sys.exit(1);

X=np.load(sys.argv[1]);  X=X[X.files[0]];    # imágenes MNIST en npz
xl=np.load(sys.argv[2]); xl=xl[xl.files[0]]; # etiquetas MNIST en npz
trper=int(sys.argv[3]); # porcentaje para training
dvper=int(sys.argv[4]); # porcentaja para dev
seed=int(sys.argv[5]);  # semilla
tr=sys.argv[6];         # nombre ficheros tr (tr imágenes y trl labels)
dv=sys.argv[7];         # nombre ficheros dv (dv imágenes y dvl labels)
N=X.shape[0];           # número de datos

# permutamos aleatoriamente todos los datos
np.random.seed(seed); p=np.random.permutation(N); X=X[p]; xl=xl[p];

# guardamos imágenes y labels de tr y dv en formato npz
Ntr=round(trper/100*N); Ndv=round(dvper/100*N);
np.savez_compressed(tr,     dat=X[:Ntr]);
np.savez_compressed(tr+'l', dat=xl[:Ntr])
np.savez_compressed(dv,     dat=X[N-Ndv:]);
np.savez_compressed(dv+'l', dat=xl[N-Ndv:])
```

- Obtén una partición tr-dev con un 20 % y 10 % de MNIST:

```
./part.py train-images-idx3-ubyte.pca20.npz
↪  train-labels-idx1-ubyte.pca20.npz 20 10 23 tr dv
```

  - Comprueba que tienes los siguientes ficheros:

```
du -sh tr.npz trl.npz dv.npz dvl.npz
```

```
1,8M tr.npz
12K  trl.npz
900K dv.npz
8,0K dvl.npz
```

  - Comprueba informalmente que estén bien:

```
>>> import numpy as np
>>> z=np.load('tr.npz')
>>> z.files
['dat']
>>> z['dat'].shape
(12000, 20)
>>> ... # haz lo mismo con trl.npz, dv.npz y dvl.npz
```

3.3. Experimento con el kernel lineal

- Script para realizar un experimento con el kernel lineal:

linear.py

```python
#!/usr/bin/env python
import sys
import numpy as np
from sklearn import svm

if len(sys.argv)!=5:
  print('Usage: %s <tr.npz> <trl.npz> <dv.npz> <dvl.npz>' % sys.argv[0]);
  sys.exit(1);

tr=np.load(sys.argv[1]);  tr=tr[tr.files[0]];
trl=np.load(sys.argv[2]); trl=trl[trl.files[0]];
dv=np.load(sys.argv[3]);  dv=dv[dv.files[0]];
dvl=np.load(sys.argv[4]); dvl=dvl[dvl.files[0]];

# normalizamos las características en [-1,1]
S=max(tr.max(),abs(tr.min())); tr/=S; dv/=S;

# probamos diferentes valores para el parámetro de penalización C, C>0,
# y hallamos el error en tr y dv para cada uno de ellos
for C in [1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4]:
  clf=svm.SVC(kernel='linear',C=C).fit(tr, trl)
  etr=(trl!=clf.predict(tr)).mean();
  edv=(dvl!=clf.predict(dv)).mean();
  print("%8g %8.2f %8.2f" % (C,etr*100,edv*100));
```

- Experimento con el kernel lineal:

```
./linear.py tr.npz trl.npz dv.npz dvl.npz >linear.out &
```

  - Examinamos el resultado con cat o tail -f:

```
tail -f linear.out
```

linear.out

```
    0.001     88.61     88.82
     0.01     20.20     19.82
      0.1     12.04     11.82
        1     10.16     10.30
       10      9.34     10.10
      100      9.02     10.17
     1000      9.10     10.18
    10000      9.07     10.18
```

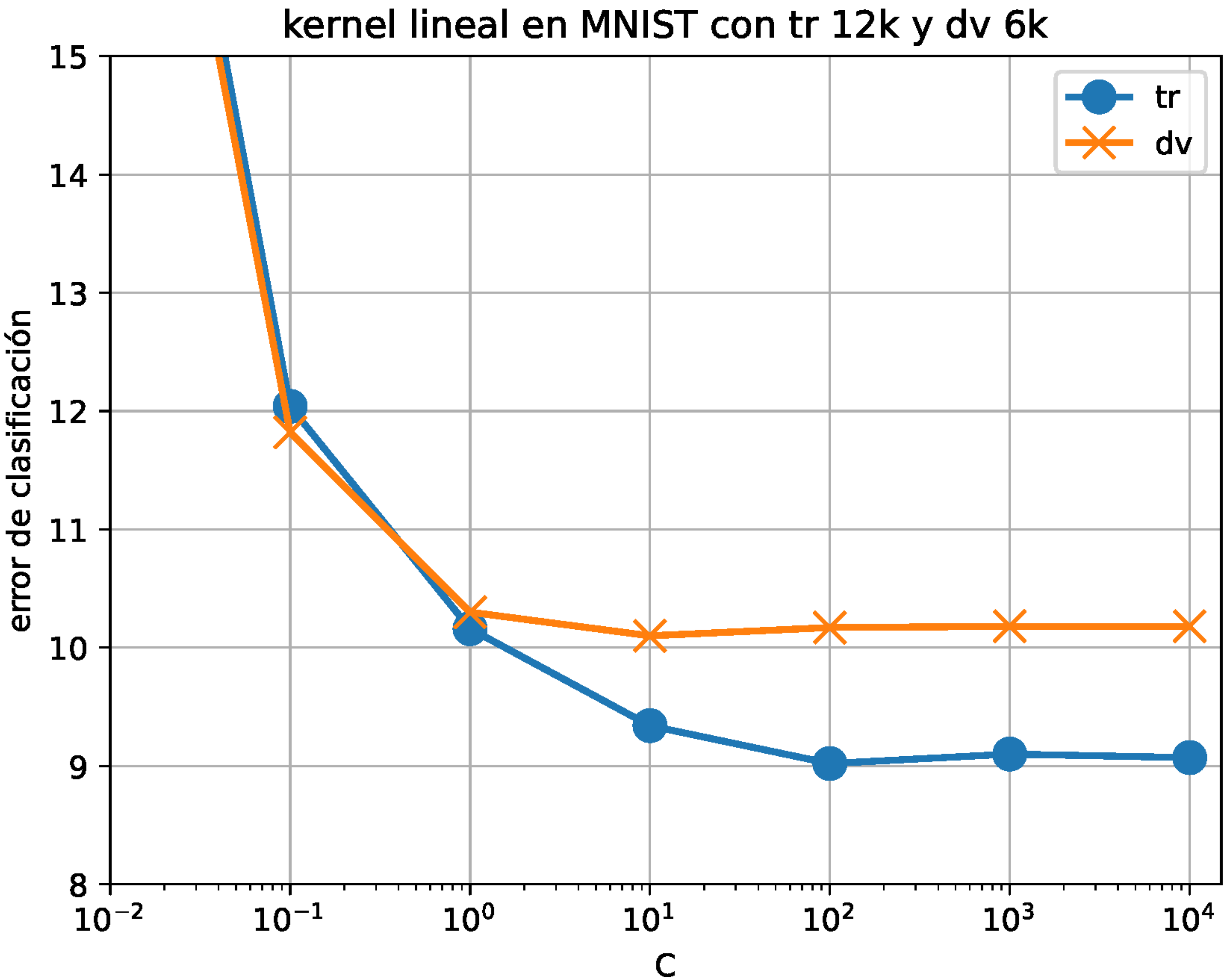
  - Observamos que el error en tr y dv decrece al aumentar C hasta 10; luego se estabiliza tanto en tr (9.1) como en dv (10.1)
  - La debilitación de restricciones ($C \to 0^+$) no parece ayudar
  - Con restricciones fuertes ($C \to \infty$) obtenemos un error cercano al $10\,\%$, lo cual parece indicar que las características originales no son suficientes para "linearizar" los datos

- Script para hacer una gráfica con los resultados:

linear_plot.py

```python
#!/usr/bin/env python
import numpy as np
import matplotlib.pyplot as plt

d=np.loadtxt('linear.out')
fig, ax = plt.subplots()
ax.set_title('kernel lineal en MNIST con tr 12k y dv 6k');
ax.grid();
ax.set_xscale('log');
ax.set_xlim([1e-2,1.5e4]);
ax.set_xlabel('C');
ax.set_ylim([8,15]);
ax.set_ylabel('error de clasificación');
ax.plot(d[:,0], d[:,1], label='tr', lw=2, marker='o', markersize=10)
ax.plot(d[:,0], d[:,2], label='dv', lw=2, marker='x', markersize=10)
ax.legend();
plt.savefig('linear.pdf');
plt.show();
```



kernel lineal en MNIST con tr 12k y dv 6k
error de clasificación
15
14
13
12
11
10
9
8
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
C
tr
dv

- Script para entrenar con todos los datos y guardar el modelo:

linear_save.py
```python
#!/usr/bin/env python
import numpy as np
from sklearn import svm
import pickle

tr=np.load('train-images-idx3-ubyte.pca20.npz');  tr=tr[tr.files[0]];
trl=np.load('train-labels-idx1-ubyte.pca20.npz'); trl=trl[trl.files[0]];
S=max(tr.max(),abs(tr.min())); tr/=S;
C=10;
clf=svm.SVC(kernel='linear',C=C).fit(tr,trl);
etr=(trl!=clf.predict(tr)).mean();
print("%8g %8.2f" % (C,etr*100));
pickle.dump(clf,open('linear_save.clf','wb'));
```

```
./linear_save.py >linear_save.out &
```

linear_save.out
```
      10      9.60
```

```
du -sh linear_save.clf
```

```
3,4M	linear_save.clf
```

▶ Script para comprobar que el modelo guardado está bien:

linear_check.py

```python
#!/usr/bin/env python
import numpy as np
from sklearn import svm
import pickle

tr=np.load('train-images-idx3-ubyte.pca20.npz');  tr=tr[tr.files[0]];
trl=np.load('train-labels-idx1-ubyte.pca20.npz'); trl=trl[trl.files[0]];
S=max(tr.max(),abs(tr.min())); tr/=S;
clf=pickle.load(open('linear_save.clf','rb'));
etr=(trl!=clf.predict(tr)).mean();
print("%8g %8.2f" % (clf.C,etr*100));
pickle.dump(clf,open('linear_save.clf','wb'));
```

```
./linear_check.py >linear_check.out &
```

linear_check.out

```
      10      9.60
```

4. Ejercicio

- Repite el experimento con el kernel lineal haciendo uso de los otros tres kernels disponibles: poly, rbf y sigmoid

- Los principales parámetros a explorar son $C \geq 0$ y $\gamma > 0$
  - poly también depende de d (degree) y r (coef0)
  - sigmoid también depende de r (coef0)

- Los resultados con cada kernel deberán presentarse mediante gráficas apropiadas que faciliten su interpretación, siendo de especial interés identificar qué parámetros y valores tienen mayor efecto en el grado de subajuste o sobreajuste de los modelos
  - *Ejemplo:* un modelo entrenado con C muy pequeño debilitará mucho las restricciones, por lo que se hallará fuertemente regularizado y posiblemente subajustado (en dv y también en tr)

- Elabora una memoria en pdf que describa los experimentos realizados y resultados obtenidos; en particular, por cada uno de los tres kernels a probar, la memoria debe:
  - Describir los scripts experimentales empleados
  - Incluir (al menos) una gráfica con los resultados obtenidos
  - Discutir los resultados de la gráfica
  - Proporcionar los valores de los parámetros escogidos para entrenar el modelo final con todos los datos, así como una estimación de su error (con tr)
- Sube la memoria y el modelo final más prometedor (de los tres obtenidos) a la tarea Poliformat de la práctica
  - Tanto la memoria como el modelo pueden deben subirse comprimidos (p.e. con gzip -9); el modelo no debe pasar de 4Mb
  - Si la práctica se hace en pareja, basta que la entregue una persona e indique claramente el nombre de la otra