UNIVERSIDAD CATÓLICA DE SAN PABLO
MAESTRÍA EN CIENCIA DE LA COMPUTACIÓN
SISTEMAS INTELIGENTES

Máquina de Vectores de Soporte (SVM) (CORRECCIÓN) Prof. Graciela Meza Lovón

Palomino Paucar, Daniel Alfredo
Noviembre 26, 2018
https://bit.ly/2r7y556

1. PREGUNTAS DE TEORÍA

Sea el conjunto $N = ((1,6),-1),((4,9),-1),((4,6),-1),((5,1),1),((9,1),1),((0,3),1),((2,2),-1),((3,1),-1)$ **y el hiperplano** H_1 **definido anteriormente.**

9. Calcule la holgura de los ejemplos no separables.

Respuesta:
Dado que para los ejemplos no separables:

$$\epsilon_i = 1 - y_i D(x_i)$$

Se tendrá para los ejemplos no separables $((0,3),1),((2,2),-1),((3,1),-1)$:

1. $((0,3),\ 1)$: $\epsilon_i = 3{,}8284$
2. $((2,2),-1)$: $\epsilon_i = 0{,}2929$
3. $((3,1),-1)$: $\epsilon_i = 1{,}7071$

2. Preguntas de Investigación

3. Implementación

2. Experimente y muestre resultados usando diferentes valores para los parámetros de los kernels: lineal, polinomial, gaussiano, y el parámetro C. Los resultados deben ser mostrados en el documento pdf.

Implementación:

Como no se ha incluído el dataset, he propuesto trabajar el presente ejercicio con el siguiente dataset (Iris Flower): `https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data`

Lectura de Dataset

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-databases"
      "/iris/iris.data"
# Asignar columnas al Dataset para su posterior lectura
colnames = ['sepal-length', 'sepal-width',
            'petal-length', 'petal-width', 'Class']
# Lectura del Dataset usando pandas
irisdata = pd.read_csv(url, names=colnames)
```

Exploración de Datos

```
# Descripci n del Dataset
bankdata.head()
```

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Preprocesamiento

```
# Separaci n de las columnas del dataset para crear los valores de entrada
# y el valor de salida
X = irisdata.drop('Class', axis=1)
y = irisdata['Class']
# Separaci n del dataset en valores de entranemiento y testeo
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Lineal con C=0.5
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: linear Y par metro C=0.5
svclassifier = SVC(kernel='linear', C=0.5)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

Lineal con C=1.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: linear Y par metro C=1.0
svclassifier = SVC(kernel='linear', C=1.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  1  8]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	0.93	1.00	0.96	13
Iris-virginica	1.00	0.89	0.94	9
avg / total	0.97	0.97	0.97	30

Lineal con C=2.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: linear Y par metro C=2.0
svclassifier = SVC(kernel='linear', C=2.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

Lineal con C=5.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: linear Y par metro C=5.0
svclassifier = SVC(kernel='linear', C=5.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  1  8]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	0.93	1.00	0.96	13
Iris-virginica	1.00	0.89	0.94	9
avg / total	0.97	0.97	0.97	30

Polinomial con C=0.5
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: poly Y par metro C=0.5
svclassifier = SVC(kernel='poly', degree=8, C=0.5)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

Polinomial con C=1.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: poly Y par metro C=1.0
svclassifier = SVC(kernel='poly', degree=8, C=1.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

Polinomial con C=2.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: poly Y par metro C=2.0
svclassifier = SVC(kernel='poly', degree=8, C=2.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         9

    avg / total       1.00      1.00      1.00        30
```

Polinomial con C=5.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: poly Y par metro C=5.0
svclassifier = SVC(kernel='poly', degree=8, C=5.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

Gaussiano con C=0.5
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: rbf y C=0.5
svclassifier = SVC(kernel='rbf', C=0.5)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

Gaussiano con C=1.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: rbf y C=1.0
svclassifier = SVC(kernel='rbf', C=1.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  1  8]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	0.93	1.00	0.96	13
Iris-virginica	1.00	0.89	0.94	9
avg / total	0.97	0.97	0.97	30

Gaussiano con C=2.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: rbf y C=2.0
svclassifier = SVC(kernel='rbf', C=2.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

Gaussiano con C=5.0
Entrenamiento

```
from sklearn.svm import SVC
# Elecci n del Kernel: rbf y C=5.0
svclassifier = SVC(kernel='rbf', C=5.0)
# Ejecuci n del entrenamiento
svclassifier.fit(X_train, y_train)
```

Predicción y Validación

```
# Predicci n usando los valores de testeo
y_pred = svclassifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
# Creaci n de la matriz de confusi n
print(confusion_matrix(y_test,y_pred))
# Creaci n de reporte de resultados
print(classification_report(y_test,y_pred))
```

```
[[ 8  0  0]
 [ 0 13  0]
 [ 0  0  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	8
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	9
avg / total	1.00	1.00	1.00	30

3. Dentro de la sección de Implementación incluya una subsección donde indique las instrucciones para ejecutar el código.

Implementación:

La presente implementación ha sido realizada en python 3.6 usando anaconda y jupyter-lab como herramientas de virtualización y visualización.

El notebook desarrollado que incluye todos los pasos descritos, así como los datasets, ha sido publicado en mi cuenta de github:

`https://github.com/dpalominop/SistemasInteligentes/blob/master/semana7/homework/svm.ipynb`