IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

UETAF IML

Practical on SVMs

Yannis Haralambous (IMT Atlantique)

October 20, 2021

In this practical we will first scrutinize an elementary example of SVM and then see some SVM applications under Python. The packages used are `scikit-learn` (and its subpackages `svm` et `datasets`), `numpy` and `matplotlib`.

1 An elementary SVM

In the Euclidean plane $\mathbb{R}^2$ take individuals $(1, 1)$, $(1, 5)$, $(5, 1)$ and $(5, 5)$, assigned to classes $-1, 1, 1, 1$.

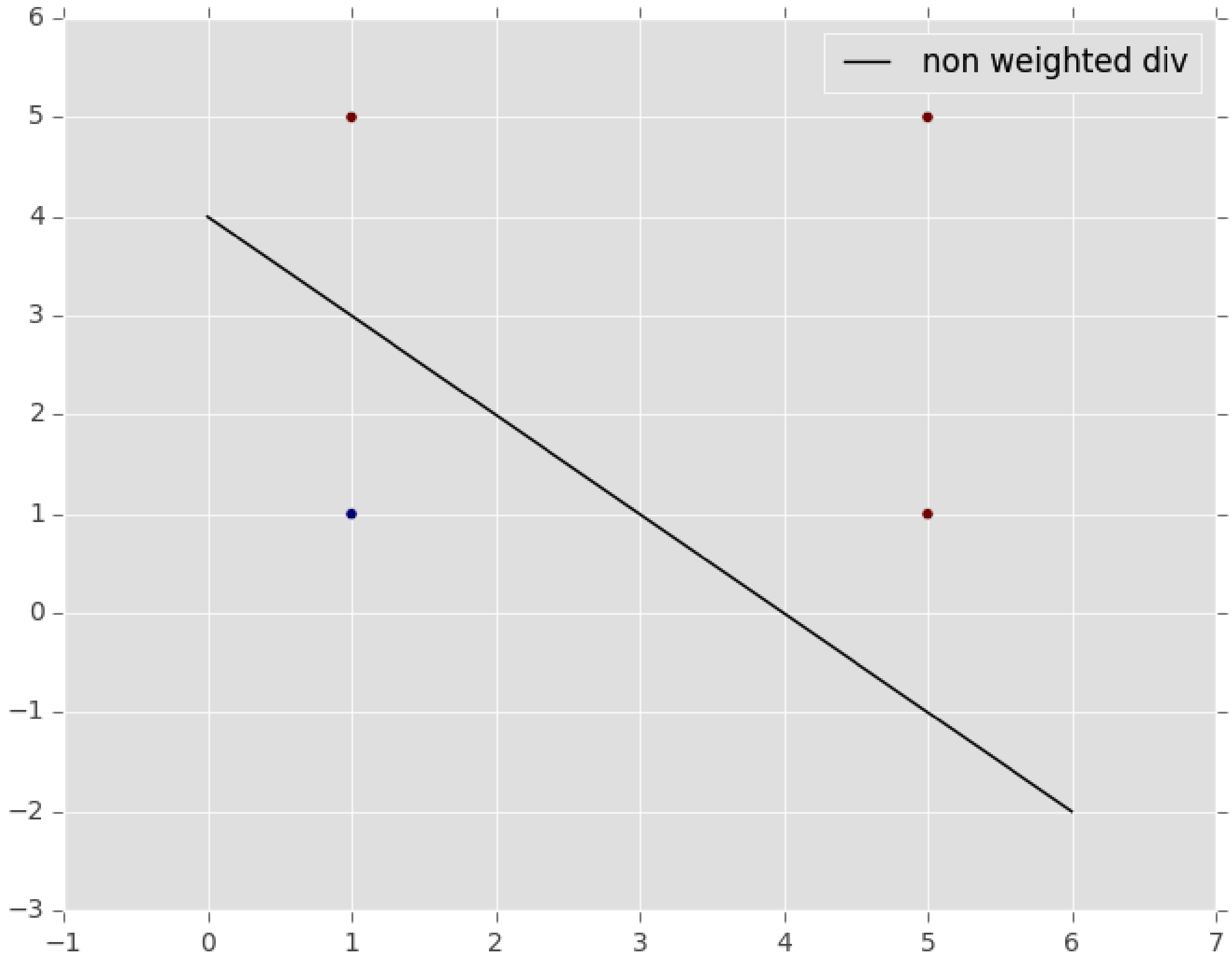
1) Manually calculate the vector (a, b) and the value of c of the maximal margin classifier straight line.

2) Implement:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.style.use("ggplot")
from sklearn import svm
X = np.array([[1, 1], [1, 5], [5, 1], [5, 5]])
y = [-1,1,1,1]
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
print("clf.support_vectors_ = ")
print(clf.support_vectors_)
print(f"{clf.n_support_ = !s}")
print(f"{             w = !s}")
w = clf.coef_[0]
print(w)
a = -w[0] / w[1]
xx = np.linspace(0,6)
yy = a * xx - clf.intercept_[0] / w[1]

fig, ax = plt.subplots()
ax.plot(xx, yy, "k-", label="non weighted div")
ax.scatter(X[:, 0], X[:, 1], c=y)
ax.legend()
```

to obtain



What is the meaning of the obtained results:

```
clf.support_vectors_ =
[[1. 1.]
 [1. 5.]
 [5. 1.]]
clf.n_support_ = [1 2]
             w = [0.49975586 0.49975586]
```

2 Iris

We will start with the famous data set “Iris”: the size in centimeters of petals and other parts of some flowers. For every individual we have four number and the class, among*Iris setosa*, *Iris versicolor* and *Iris virginica*. We have 150 individuals and equidistributed classes.

Do

```
from sklearn import svm
from sklearn import datasets
from sklearn.model_selection import train_test_split
# Load data
iris = datasets.load_iris()

# Extract arrays
X, y = iris.data, iris.target
```

```
# Extract some useful information
num_classes = len(iris.target_names)
classes_labels = sorted(set(iris.target))

# Initialize classfier
clf = svm.SVC()
```

Out of the `iris` data create a training set `X_train,y_train` with 100 random individuals and a test set `X_test,y_test` with what remains. *Advice: use the `shuffle` function from package `random`.*

Launch the SVM by writing

```
clf.fit(X_train, y_train)
```

For predictions, use

```
y_pred = clf.predict(X_test)
```

Calculate, using Python list comprehension the precision and recall of each class.

Check result against `sklearn.metrics.classification_report`.

By using class `KFold` from package `sklearn.cross_validation`, write the code a 10-crossed validation and obtain average precision and recall for each class.

Use various kernel types and parameter values to see how the performances vary.

You have the choice between four kernels (option `kernel` of `SVC`) :

1. linear `linear` $k(x,x') = \langle x, x'\rangle$;
2. polynomial `pol` $k(x,x') = (\gamma \cdot \langle x, x'\rangle + r)^d$;
3. radial `rbf` $k(x,x') = e^{-\gamma\|x-x'\|}$ (par défaut) ;
4. sigmoid `sigmoid` $k(x,x') = \mathrm{th}(\gamma \cdot \langle x, x'\rangle)$.

Parameters γ, d and r are written `gamma`, `degree` and `coef0`, resp. The cost parameter C (see in the class) is written `cost`.

3 Chronic Kidney Disease

3.1 Data preparation

- Fetch the file `chronic_kidney_disease_full.arff` from Moodle. Read the introductory comments and get acquainted with the ARFF data format.

  (For more information see `http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease`)
- Read the data into a `pandas.DataFrame` named `ckd`
- Clean up the data
- Decode strings: `pandas.DataFrame.columns`, `pandas.DataFrame.dtypes`
- Handle nodata values (replace them with actual `np.nan`): `pandas.DataFrame.replace`.
- Convert data types that are still wrong: `pandas.DataFrame.apply`, `pandas.to_numeric`.
- Convert categorical features to corresponding data type: `pandas.DataFrame.astype`.
- Remove the "ground-truth" column and store its values aside: `pandas.DataFrame.drop`
- For missing numeric values, replace with the feature average: `pandas.DataFrame.fillna`

- For missing categorical values, replace with the feature most occuring value `pandas.DataFrame.mode` (hint: `ckd.mode().iloc[0]`)
- One-hot encode categorical features: `pandas.get_dummies`
- Normalize the data
- Get acquainted with the input format of SVM Light, convert the data into that format, save them in a file `data.dat`.
- Convert the "ground-truth" data into the right numeric format into a numpy array `y`: `pandas.Series.map`, lambda expressions, `pandas.Series.to_numpy`
- Extract the data in a numpy array `X` `pandas.DataFrame.to_numpy`
- Write data.dat from `X` and `y` `str.join`, f-strings `open`, `io.TextIOBase.write`

3.2 Data preparation for cross validation

Read `data.dat` and shuffle lines. Divide in ten parts, put every test corpus into `test`*`i`*`.dat` for $i = 0, \ldots, 9$, and every training corpus into `train`*`i`*`.dat` for $i = 0, \ldots, 9$.

3.3 Cross-validation and results

Download and install SVM Light from `https://www.cs.cornell.edu/people/tj/svm_light/index.html` Find out the command line syntax and run SVM Light ten times from within Python on the `test_xx.dat` and `train_xx.dat` files. Capture the data returned from SVM Light, and parse it to extract runtime, error, precision and recall `subprocess.run` `subprocess.CompletedProcess.stdout`, `bytes.decode` `re.match`.

Calculate the average runtime, error, precision and recall. Use various kernels and play with the parameters to see whether the results can be improved, without affecting performance too much.

Compare with scikit-learn

4 SPAM

For those who finished the previous exercise and are still motivated for another example, which they can continue at home, fetch the SPAM dataset from Hewlett-Packard:

`https://archive.ics.uci.edu/ml/datasets/Spambase`

It contains 4 061 individuals with 57 features and the class (SPAM or not SPAM). There are 39,4% SPAM individuals and 60,59% not SPAM individuals. It is preferable to avoid false positives, see if you can get zero false positives and check how many SPAMs pass the filter then.