- — Apply Kernel SVM on synthetic and real datasets
- — Investigate the choice of hyper-parameter C
- — Extend to multi-class classification problem
- — *Provided codes* : functions included in `utility_svm.py` on Moodle.

This Session relies on the same materials as for Linear SVM Session. You should refer to it.

1 Synthetic data

Let consider a non-linear binary classification problem with class. The samples we will deal with are shown in Figure 1. We will rely on non-linear kernel SVM to classify these data.

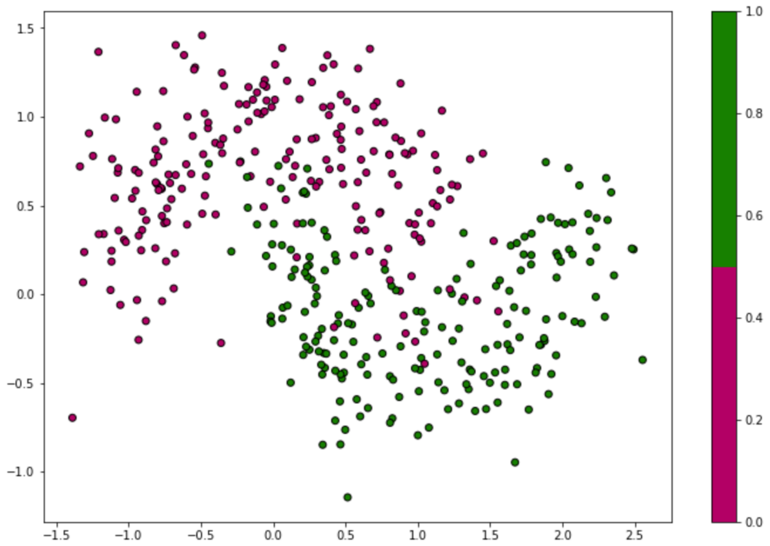

FIGURE 1 – Two moons dataset

1. Generate $n = 900$ two-moons samples

```
from sklearn.datasets import make_moons
n = 900
X, Y = make_moons(n_samples=n, noise=0.25, random_state=42)
```

2. Split the data into respectively training (`Xtrain, Ytrain`), validation (`Xval, Yval`), and test sets (`Xtest, Ytest`) of equal size.
3. Visualize the training samples and check that you will need a non-linear decision function.

```
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
```

```
cm = plt.cm.PiYG
cm_bright = ListedColormap(["#b30065", "#178000"])
plt.figure(figsize=(12,8))
plt.scatter(X_train[:, 0], X_train[:, 1], c=Y_train, cmap=cm_bright)
```

4. Let design a non-linear SVM.
   (a) Let define the non-linear SVM using a rbf (also termed as gaussian) kernel $k(\mathbf{x}, \mathbf{z}) = \exp^{-\gamma\|\mathbf{x}-\mathbf{z}\|^2}$. We arbitrary set $\gamma = 0.1$ and the hyper-parameter C of SVM as $C = 10$.

   ```
   from sklearn.svm import SVC
   clfker = SVC(kernel="rbf") # SVM with rbf kernel
   # set parameters gamma and C
   clfker.gamma = 0.1
   clfker.C = 10
   ```

   (b) The model being specified, let learn its parameters (the α_i). Plot the decision frontier and comment the obtained curve. Is is satisfactory ?

   ```
   clfker.fit(X_train, Y_train)
   # Plot decision frontier
   from utility_svm import plot_regions_decision_2d
   plot_decision_regions_2d(X_train, Y_train, clfker, 0.02, title="
       Kernel SVM")
   ```

5. We aim to analyzing the influence of the kernel parameter γ. For the sake, vary γ in $\{10^{-2}, 10^{-1}, 1, 10, 100\}$, fit the corresponding SVM, visualize and comment on how the decision frontier changes. Especially for small and large values of γ justify the shape of the decision frontier.

6. To tune appropriately our SVM, we require to set the ”optimal” values of C and γ. As for the previous practical sessions, let apply the validation procedure. Select C in the logarithmic range $\{10^{-3}, \cdots, 10^2\}$ and γ in $\{10^{-2}, \cdots, 10^2\}$ For each pair of (C, γ), train a SVM model, compute the error rate on validation set.

   ```
   from sklearn.metrics import accuracy_score
   # Ranges of C and Gamma
   vectC = np.logspace(-3, 2, 6)
   vectGamma = np.logspace(-2, 2, 6)
   err_val = np.empty((vectC.shape[0], vectGamma.shape[0]))

   for ind_C, C in enumerate(vectC):
     clfker.C = C
     for ind_gam, paramKer in enumerate(vectGamma):
       clfker.gamma = paramKer
       clfker.fit(X_train, Y_train)
       err_val[ind_C, ind_gam]= 1 - accuracy_score(Y_val, clfker.predict(
           X_val))
   ```

   Plot the obtained error rates. What is the optimal pair $(C_{\text{opt}}, \gamma_{\text{opt}})$ to select ?

```
plt.imshow(err_val, extent=[min(vectC),max(vectC),min(vectGamma),max(
    vectGamma)],aspect="auto");
plt.colorbar()
plt.xlabel("C"); plt.ylabel("gamma");
plt.title("Validation error rate")
plt.show()
```

$(C_{\text{opt}}, \gamma_{\text{opt}})$ are retrieved as corresponding to the minimal error rate.

```
ind_C, ind_gamma = np.unravel_index(np.argmin(err_val), err_val.shape)
Copt = vectC[ind_C]
GammaOpt = vectGamma[ind_gamma]
```

7. Thereon, train your optimal kernel SVM and evaluate its performance either on training or test set. Visualize the decision border and comment the results.
8. Repeat questions 4-7 for the polynomial kernel $k(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^\top \mathbf{z})^{\text{degree}}$. Vary `degree` in $\{1, 2, 3, 4, 5, 6, 7\}$. Beware to set coef0 $= 1$ in SVC. The polynomial kernel SVM has to be defined as

```
  SVC(kernel="poly", gamma="scale" , coef0=1)
```

   How the results of SVM with polynomial kernel compare to SVM based on rbf kernel ?

2 Spam classification

In this exercise we will compare kernel SVM with its linear counterpart on the spam classification problem. Refer to the Linear SVM practical session for the dataset details. `spambase.data` and the features name `spambase_variables.csv` are available on Moodle.

1. Read the files and extract the inputs X and the output Y (last column in the dataset).
2. Split the data into training and test sets. The test set size should be $1/3$ of the data.
3. The goal is to learn a non-linear spam classifier. Design an rbf kernel SVM and evaluate its performances on test set. Highlight the hyper-parameters selection and all the important steps of the model learning and assessment.

   Compare to the linear SVM results.