Pattern Classification And Machine Learning - EPFL - Fall 2015
Benoit Seguin, Victor Kristof, and Emtiyaz Khan

1. Support Vector Machines

1.1 Goals

The goal of this exercise is to:

- Understand the limitations of linear models.
- Understand Support Vector Machines (SVM).
- Play with the parameters of SVM and understand their effect on the performance.

1.2 Two-spirals dataset

We will use the data shown in Fig. 1.1). The main code you will work with is in `main.m`. Our goal is to train a binary classifier on this dataset.

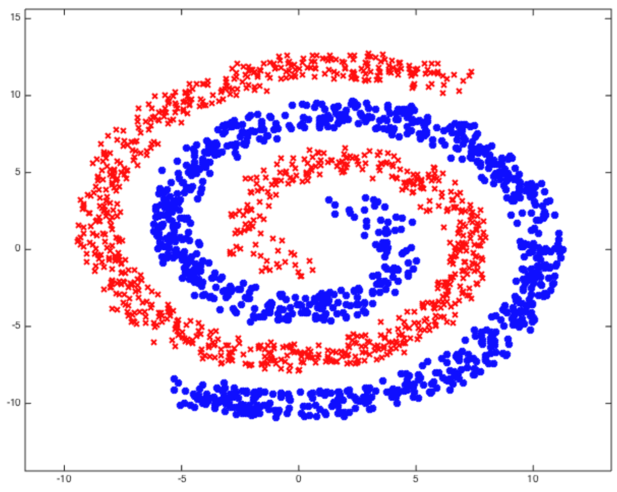

Figure 1.1: Two-spirals dataset.

1.3 Classification with linear model

Exercise 1.1 We will first understand why logistic regression does not work on this dataset.

- First, run `main.m`. This will plot the data as shown in Fig. 1.1.
- Now, think about the following: What kind do decision boundaries logistic regression will give?
- Apply penalized logistic regression and visualize the result. Refer to the lab on logistic-regression for visualization. Note that the labels are in $\{-1, 1\}$, so you might want to convert it to $\{0, 1\}$.

■

1.4 Linear SVM

We will now implement a *linear* classfier using SVM. We will start with a linear classifier, so that we can compare with logistic regression. We will start with the code given in the section marked “Exercise 1.2” in `main.m`, which applies SVM to the `weights_heights.mat` data (we used this data during the lab on logistic regression).

The implementation of SVM is based on the Sequential Minimal Optimization (SMO) algorithm which solves the dual problem (i.e. finding the α's from the kernel matrix K). Revise the lecture notes on SVM. We have given you the SMO code (see `SMO.m`). The algorithm works as follows: recursively select a pair of elements (α_1, α_2) in a greedy fashion, and then perform a 1-D optimization of the objective function on this pair. It is a good exercise to have a look at the code to identify these steps in the given SMO code.

Exercise 1.2 Applying SVM.

- Implement a function `linear_kernel` that computes the kernel matrix: $K = \Phi\Phi^T$.
- Get a solution for the dual problem with `[alphas, beta0] = SMO(K, ... y, C)`. You can set $C = 0.1$ for now.
- You can plot the output $\boldsymbol{\alpha}$ using `stem` function in Matlab. How many support vectors do you get ? Identify the bound and essential support vectors.
- Compute the predictions with $\hat{y} = \Phi_{test}\Phi_{train}^T\tilde{\alpha} + \beta_0$ using the code below. Here the j'th entry of $\tilde{\boldsymbol{\alpha}}$ is equal to $y_j\alpha_j$.

```
SV_inds = find(alphas>0);
X_SV = X_norm(SV_inds, :);
y_SV = y(SV_inds, :);
alphas_SV = alphas(SV_inds);
kernel_pred = linear_kernel(tX_pred, X_SV);
pred = kernel_pred * (alphas_SV .* y_SV) + beta0;
```

  Note that, since the support vectors are sparse, the above code can be optimized.
- Visualize the predictions, margin, and the support vectors on the plot.
- Try different values of C. How does it change the results ?
- How would you compute the optimal β^*?

■

1.5 A wild kernel appears!

In the linear case, the kernel matrix $K_{i,j} = \langle\Phi_i, \Phi_j\rangle$ (where $\langle\cdot,\cdot\rangle$ is a scalar product and Φ_i the feature representation of sample i) is directly obtained with $K = \Phi\Phi^T$. But one can use a different function to replace the Euclidean scalar product. A very popular choice of kernel is the Radial Basis Function (RBF):

$$RBF(\Phi_i, \Phi_j) = \exp\left(-\gamma||\Phi_i - \Phi_j||^2\right)$$

Exercise 1.3 Now, implement SVM with a non-linear kernel.

- Implement a function `rbf_kernel` and compute K with it. The rest of the code should stay the same. You can set $\gamma = 1$ and $C = 1$, for now.
- Did your predictions change?
- Run your code on the spiral dataset. You should get Figure 1.2.

- Play with different values of γ and C. How does it influence the results?
- Is it still possible to compute β^*? Why?

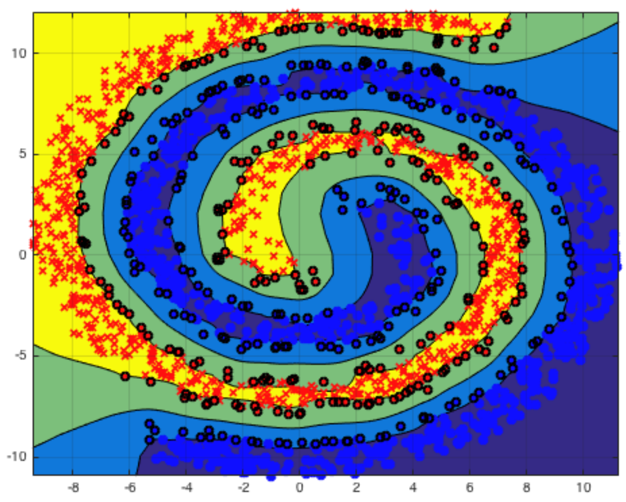

Figure 1.2: Two-spirals classification.

1.6 Bonus : Going further

So far, we gave you the values of the SVM hyperparameters and a toy dataset. How would you do proceed with in real situation, where you have to find the correct C, γ and kernels on a real dataset ?

Exercise 1.4 Real-life use of SVM.

- How can you automatically find the parameters γ and C ? Implement it. (*Hint:* they are hyperparameters).
- Apply SVM on your Project 1 classification dataset.