ENS Lyon
2023/2024 **Titouan Vayer**

Homework 1 : Kernels for ML

You have three weeks to do this homework: it must be return by Wednesday, October 16.

- You can do it by group of 2/3.
- Send the code to titouan.vayer@inria.fr with the header "Homework 1 Name 1 Name 2 Name 3".
- For the maths send a scan by mail or give it by hand on 16th october.

- Exercise 1: Coding a support vector machine algorithm ($\approx$ 6h). -

This exercise is long: focus on the practical part from question (v), the theoretical part can be done later on. Let $(\mathbf{x}_i, y_i)_{i \in [\![n]\!]}$ be a dataset with $\mathbf{x}_i \in \mathcal{X}, y_i \in \{-1, +1\}$. Let $\mathcal{H}$ be a RKHS with reproducing kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. We consider the SVM problem with the Hinge loss

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}}^2 . \tag{1}$$

(i) By using the representer theorem, show that in order to solve (1), it suffices to solve the optimization problem

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i [\mathbf{K}\boldsymbol{\theta}]_i) + \lambda \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta} , \tag{2}$$

where $\mathbf{K} = (\kappa(\mathbf{x}_i, \mathbf{x}_j))_{(i,j) \in [\![n]\!]^2}$ is the Gram matrix.

The goal now is to reformulate the objective and to find the dual optimization problem of (2). The dual will be then solved with a simple projected gradient algorithm. Suppose that we define $\xi_1, \cdots, \xi_n$ with $\xi_i \geq 0$ as

$$\xi_i = \max(0, 1 - y_i [\mathbf{K}\boldsymbol{\theta}]_i) . \tag{3}$$

Then, by definition, for any $i \in [\![n]\!], y_i [\mathbf{K}\boldsymbol{\theta}]_i \geq 1 - \xi_i$. Thus, by introducing slack variables $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_n)$, it is not too complicated to see that (2) is equivalent to

$$\min_{\substack{\boldsymbol{\theta} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^n_+ \\ \forall i \in [\![n]\!], y_i [\mathbf{K}\boldsymbol{\theta}]_i \geq 1 - \xi_i}} \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta} . \tag{4}$$

This is a convex constrained optimization problem that can be solved by solving its dual problem. To find it we look at the Lagrangian of the problem, which is, for $\boldsymbol{\alpha} \in \mathbb{R}^n_+, \boldsymbol{\beta} \in \mathbb{R}^n_+$,

$$L(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \boldsymbol{\theta}^\top \mathbf{K} \boldsymbol{\theta} - \sum_{i=1}^{n} \alpha_i \xi_i - \sum_{i=1}^{n} \beta_i (y_i [\mathbf{K}\boldsymbol{\theta}]_i - 1 + \xi_i) . \tag{5}$$

The variable $\boldsymbol{\alpha}$ accounts for the non-negativity constraint on $\boldsymbol{\xi}$ and the $\boldsymbol{\beta}$ variable accounts for the constraint $\forall i \in [\![n]\!], y_i [\mathbf{K}\boldsymbol{\theta}]_i \geq 1 - \xi_i$ (remember your optimization class, you can have a look at Section 8.6 in `https://mathurinm.github.io/assets/2022_ens/class.pdf` if needed).

(ii) Show that the minimization of the Lagragian with respect to the primal variables $\boldsymbol{\theta} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^n$ gives the conditions

$$\begin{aligned} &\forall i \in [\![n]\!], \ \theta_i = \frac{1}{2\lambda} \beta_i y_i \\ &\forall i \in [\![n]\!], \ \alpha_i + \beta_i = \frac{1}{n} . \end{aligned} \tag{6}$$

(iii) Deduce that the dual problem writes

$$\max_{\substack{\boldsymbol{\beta}\in\mathbb{R}^n \\ \forall i\in[\![n]\!], \beta_i\in[0,\frac{1}{n}]}} \sum_i \beta_i - \frac{1}{2\lambda}\sum_{ij} K_{ij}\beta_i\beta_j y_i y_j \,. \tag{7}$$

(iv) We set $g(\boldsymbol{\beta}) = \frac{1}{2\lambda}\sum_{ij} K_{ij}\beta_i\beta_j y_i y_j - \sum_i \beta_i$. The dual problem is thus equivalent to

$$\min_{\substack{\boldsymbol{\beta}\in\mathbb{R}^n \\ \forall i\in[\![n]\!], \beta_i\in[0,\frac{1}{n}]}} g(\boldsymbol{\beta}) \,. \tag{8}$$

Show that it is a convex optimization problem.

We recall that an optimization problem of the form $\min_{\boldsymbol{\beta}\in C} g(\boldsymbol{\beta})$ where g is convex and C is a convex set can be tackled with a projected gradient descent algorithm. More precisely, if $\Pi_C(\mathbf{y}) = \operatorname{argmin}_{\mathbf{x}\in C} \ \|\mathbf{x}-\mathbf{y}\|_2$ denotes the Euclidean projection of $\mathbf{y}$ onto C, the algorithm writes

$$\begin{aligned}&\beta_0 \in \mathbb{R}^n, \\ &\text{for } k \geq 0, \ \ \beta_{k+1} = \Pi_C(\beta_k - \eta\nabla g(\beta_k))\,,\end{aligned} \tag{9}$$

for a step-size size parameter $\eta > 0$. For the set $C = \{\boldsymbol{\beta}\in\mathbb{R}^n : \forall i \in [\![n]\!], \beta_i \in [a,b]\}$ we recall that

$$\Pi_C(\mathbf{y}) = \big(\max(\min(y_1, b), a), \cdots, \max(\min(y_n, b), a)\big)\,. \tag{10}$$

The interpretation is quite simple: if you are above b shrink it to b and if you are below a increase it to a, otherwise do not change the value.

(v) Based on the previous answer write a function in `python` that solves the dual problem (8). It must take as input λ, a step-size η, labels $\mathbf{y}$, a Gram matrix $\mathbf{K}$ and a stopping criterion. You can choose $\beta_0 = 0$ for simplicity.

Once the optimization problem is solved you get parameters $\boldsymbol{\beta}^\star$. With the conditions (6) the optimal $\boldsymbol{\theta}^\star$ is defined by $\forall i \in [\![n]\!], \theta_i^\star = \frac{1}{2\lambda}\beta_i^\star y_i$. From the representer theorem, the optimal $f^\star$ used for classification is then $f^\star = \sum_{i=1}^n \theta_i^\star \kappa(\cdot, \mathbf{x}_i)$. A classification rule is just given by $\operatorname{sign}(f^\star)$.

(vi) The goal now is to compare with the `scikit-learn` implementation. The SVM used in classification can be found in `sklearn.svm.SVC`. To compare both we will use the following $d = 2$ dataset.

```
from sklearn.model_selection import train_test_split
from sklearn import datasets
X, y = datasets.make_moons(n_samples=500, random_state=42, noise=0.1)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.5, shuffle=True
)
```

First plot the dataset with `matplotlib` and change the labels to match $\{+1, -1\}$ (by default the classes are $\{+1, 0\}$). For the rest of the practical exercise we will use the Gaussian kernel $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|_2^2)$.

(vii) Based on question (v) create a SVM classifier in `python` that takes as input a dataset $\mathbf{X}$, labels $\mathbf{y}$ a regularization parameter λ, and outputs the optimal SVM classifier $\operatorname{sign}(f^\star)$ where $f^\star = \sum_{i=1}^n \theta_i^\star \kappa(\cdot, \mathbf{x}_i)$.

(viii) Compare the performances on the test set of this classifier to the one of the `scikit-learn` implementation. Be careful on the influence of the hyperparameters: you must properly choose λ (you can choose λ in a grid so as to minimize the classification error on the training set for example, or do some proper cross-validation). Also be careful on the train/test split: the $\theta_i^\star$ must be found by using the training data only !

(ix) What are the performances when you use a linear kernel instead ? Can you explain why ?

Remark 1 (The correct implementation)**.** *The aim of this exercise to implement a SVM classifier. However, note that the optimization problem tackled is slightly different than the "traditional" SVM classifier that considers also a* bias term $b \in \mathbb{R}$ *and solves*

$$\min_{f \in \mathcal{H}, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(f(\mathbf{x}_i) + b)) + \lambda \|f\|_{\mathcal{H}}^2 \,. \tag{11}$$

Despite being quite similar the resulting optimization problem is quite different, and a less direct to solve (a simple projected gradient descent cannot be used).

- Exercise 2: The quadratic kernel ($\approx$ 1h) -

What is the RKHS corresponding to the kernel $\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle^2$?

- Exercise 3 (<u>bonus</u>): Positive definiteness of the Gaussian kernel ($\approx$ 1h30) -

The purpose of this exercise is to show that the Gaussian kernel $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|_2^2 / 2\sigma^2)$ is a PD kernel for any $\sigma > 0$. In the following $\kappa_1, \kappa_2, \cdots$ are fixed PD kernels.

(x) Show that $\gamma \kappa_1$ for any $\gamma > 0$ is a PD kernel.

(xi) Show that $\kappa_1 + \kappa_2$ is a PD kernel.

(xii) Suppose that $\kappa(\mathbf{x}, \mathbf{y}) := \lim_{m \to +\infty} \kappa_m(\mathbf{x}, \mathbf{y})$ exists for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Show that it defines a PD kernel.

(xiii) Consider two $n \times n$ PSD matrices $\mathbf{K}_1, \mathbf{K}_2$ and the matrix $\mathbf{K}_1 \odot \mathbf{K}_2$ defined by $\forall (i, j) \in [\![n]\!]^2$, $[\mathbf{K}_1 \odot \mathbf{K}_2]_{ij} = [\mathbf{K}_1]_{ij} [\mathbf{K}_2]_{ij}$ (this is known as the *Hadamard product* of two matrices). Show that $\mathbf{K}_1 \odot \mathbf{K}_2$ is a PSD matrix. Tips: for matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{D} = \mathrm{diag}(\mathbf{x})$ show that we have $\langle \mathbf{DA}, \mathbf{BD} \rangle_F = \mathbf{x}^\top (\mathbf{A} \odot \mathbf{B}) \mathbf{x}$.

(xiv) Deduce that $\kappa(\mathbf{x}, \mathbf{y}) := \kappa_1(\mathbf{x}, \mathbf{y}) \kappa_2(\mathbf{x}, \mathbf{y})$ is a PD kernel.

(xv) Consider $f : \mathcal{X} \to \mathbb{R}$ then show that $\kappa(\mathbf{x}, \mathbf{y}) := f(\mathbf{x}) \kappa_1(\mathbf{x}, \mathbf{y}) f(\mathbf{y})$ is a PD kernel.

(xvi) From the previous answers prove that $\kappa(\mathbf{x}, \mathbf{y}) := \exp(-\|\mathbf{x} - \mathbf{y}\|_2^2 / 2\sigma^2)$ is a PD kernel.