"Kernel methods in machine learning"
Homework 2
Due on 03/06/2024 1:30 pm

1 General instructions:

1. **The delivery must be a single PDF file containing answers to all questions.**
2. **You must upload the PDF to the GradeScope platform after creating an account there. See instructions in the course webpage for using GradeScope.**

This homework contains both mathematical and coding questions:

- All exercises are essentially coding questions. Some questions require providing answers **without proofs**. Please only include the final results in this case, without derivations.
- The coding questions require implementing some methods that are described in a Jupyter notebook (Homework.ipnb) attached to this homework. Please follow the template of the notebook and only fill in the gaps whenever asked for. You should **take a screenshot of the code** you wrote and include it to the PDF.
- Some questions require running a code block in the Jupyter notebook to check your implementation, after selecting reasonable values for some hyper-parameters. You should take a screenshot of all whole output of the (figures + any text that appears) and include it to the PDF.

Exercice 1. Support Vector Classifier

Consider a dataset of N pairs (x_i, y_i) where each x_i is a vector of dimension d and y_i is a binary class, i.e. $y_i \in \{-1, 1\}$. We would like to separate the two classes of samples with a **separating hyper-surface** of equation $f(x_i)+b = 0$ such that $f(x_i)+b \leq 0$ if x_i belongs to the class $y_i = -1$ and $f(x_i)+b \geq 0$ if $y_i = 1$. To achieve this, we consider functions f that belong to a Reproducing Kernel Hilbert Space $\mathcal{H}$ of kernel k. Such choice allows to represent highly non-linear hyper-surfaces while still solving a convex problem of the form:

$$\begin{aligned}
\min_{f,b,\xi_i} \quad & \frac{1}{2}\|f\|^2 + C\sum_{i=1}^{n}\xi_i \\
\text{s.t.} \quad & y_i(f(x_i)+b) \geq 1 - \xi_i \\
& \xi_i \geq 0
\end{aligned} \tag{1}$$

1. Without providing the details of the calculations:
   (a) Provide an expression for the Lagrangian of the problems in eq. (1) in terms of N dual parameters $\alpha_i \geq 0$ corresponding the margin inequalities and N dual parameters $\mu_i \geq 0$ corresponding to the positivity constraints on ξ_i whenever applicable.
   (b) Using the optimality condition on the Lagrangian, express the dual problem as a **constrained minimization** over $(\alpha_i)_{i\in\{1,\dots,N\}}$ and express $f(x)$ in terms of α_i and relevant quantities.
   (c) Using Strong duality (KKT conditions), find a condition characterizing the **support vector** points x_i that are on the margin of the separating hyper-surface, i.e. the points satisfying the equation $y_i(f(x_i)+b) = 1$.

2. **(a)** In the notebook, implement the method `kernel` of the classes RBF and Linear, which takes as input two data matrices X and Y of size $N \times d$ and $M \times d$ and returns a gramm matrix G of shape $N \times M$ whose components are $k(x_i, y_j) = \exp(-\|x_i - y_i\|^2/(2\sigma^2))$ for RBF and $k(x_i, y_j) = x_i^\top y_j$ for the linear kernel. (The fastest solution does not use any for loop!)
   In the notebook, the class `KernelSVC` corresponds to eq. (1):
   (b) Implement the method `fit` that computes the optimal dual parameters α_i, the parameter b and the support vectors.
   (c) Implement the method `separating_function` that takes a matrix of shape $N'\times d$ and returns a vector of size N' of evaluations of f.

(d) Report the outputs for each code block that performs a classification. What can you conclude?

Exercice 2. Kernel Ridge Regression

Given a dataset of N pairs (x_i, y_i) where x_i is a vector of dimension d and y_i is a real number. The regression is the task of finding a function f from $\mathbb{R}^d$ to $\mathbb{R}$ such that $f(x_i) + b \simeq y_i$ for some scalar b. Kernel Ridge Regression can model potentially complex/non-linear dependence between x_i and y_i by assuming the regression function f belongs to an RKHS $\mathcal{H}$ of kernel k and by solving a convex optimization problem:

$$\min_{f_j, b_j} \frac{1}{N} \sum_{i=1}^{N} \|f(x_i) + b - y_i\|^2 + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2 \tag{2}$$

When the variable y_i is a vector of dimension q a simple extension to eq. (2) consists in finding q functions $(f_j)_{1 \leq j \leq q}$ in $\mathcal{H}$ and scalars $(b_j)_{1 \leq j \leq q}$ for regressing each dimension of $(y_i)_{1 \leq j \leq d}$, i.e. $f_j(x_i) + b_j \simeq (y_i)_j$. This can be achieved by solving the problem of the form:

$$\min_{f,b} \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{q} \|f_j(x_i) + b_j - (y_i)_j\|^2 + \frac{\lambda}{2} \sum_{j=1}^{q} \|f_j\|_{\mathcal{H}}^2 \tag{3}$$

1. Without providing the details of the calculations and using the Representer theorem, provide an equivalent finite-dimensional optimization problem for both eqs. (2) and (3) and find a closed-form expression for f and b in terms of the solutions of such problems.

2. In the notebook, the classes `KernelRR` and `MultivariateKernelRR` correspond to eqs. (2) and (3):
   (a) Implement the method `fit`, which solves the finite dimensional problems obtained by the Representer theorem.
   (b) Implement the method `regression_function` that takes a matrix of shape $M \times d$ and returns a vector of size M of evaluations of f.
   (c) Report the outputs of each code block that performs a regression.

Exercice 3. Kernel PCA

One motivation for Kernel PCA is to perform non-linear dimensionality reduction of the data. This is relevant, for instance, when the data is concentrated on a lower dimensional manifold that is not a hyperplane. Given a dataset of N points x_i, the first step for performing kernel PCA is to map each data point x_i to some nonlinear feature $\varphi(x_i)$ in an RKHS $\mathcal{H}$ space corresponding to a kernel $k(x, y) = \langle \varphi(x), \varphi(y) \rangle$. One then define the centered features $\tilde{\varphi}(X_i) = \varphi(X_i) - \frac{1}{N} \sum_{j=1}^{N} \varphi(X_j)$ and the covariance operator C

$$C = \frac{1}{N} \sum_{i=1}^{N} \tilde{\varphi}(X_i) \otimes \tilde{\varphi}(X_i)$$

Where $\otimes$ denotes the tensor product associated to the inner-product $\langle ., . \rangle$, i.e. $\otimes$ is a binary operation on $\mathcal{H} \times \mathcal{H}$ such that for any u and v in $\mathcal{H}$, $u \otimes v$ is a linear operator from in $\mathcal{H}$ satisfying $(u \otimes v)f = \langle v, f \rangle u$ for any $f \in \mathcal{H}$.

Kernel PCA, consists in finding non-trivial eigenvectors of the operator C, i.e. elements $v \in \mathcal{H}$ such that $Cv = \lambda v$ for positive λ and $\|v\| = 1$.

1. Show that each non-trivial eigenvector of C can be expressed as a linear combination of the features $\tilde{\varphi}(X_i)$, with a vector of coefficients $\alpha = (\alpha_i)_{1:N}$ being an eigenvector of some square matrix G of size N and satisfying some normalization condition.

2. In the notebook, the class `Kernel_PCA` performs a Kernel PCA given some kernel as input:
   (a) Implement the method `compute_PCA` which finds the top r eigenvectors of the matrix G.
   (b) Implement the method `transform` which takes as input a data matrix of shape $M \times d$ and computes its representation of shape $M \times r$ along the r first eigenvectors of the covariance operator C.
   (c) Report the output of the code block that performs the PCA. What can you conclude?

3. **(Bonus).** The representation of the data obtained by kernel PCA can be interpreted as an r-dimensional encoding of the data (the encoder). From such encoding, it is possible to reconstruct the original data by solving a multivariate regression problem which can be interpreted as a decoder. The encoding-decoding of the data can be used

in tasks such as de-noising. In the notebook `KernelPCA`, the class `Denoiser` achieves this by making use of the classes `KernelPCA` and `MultivariateKernelRR` previously implemented.
(a) Implement the method `fit` that takes as input a noisy training set and learns both encoder and decoder.
(b) Implement the method `denoise` which takes as input a noisy test dataset and returns a corresponding de-noised dataset.
(c) Report the output of the code block that performs de-noising of a subset of MNIST digits dataset. To what extend the de-noising is successful? How can it be improved?