

“Machine Learning and Computational Statistics”

9th Homework

Exercise 1:

Consider the two-class two-dim. problem where class ω_1 (+1) consists of the vectors $\mathbf{x}_1 = [-1, 1]^T$, $\mathbf{x}_2 = [-1, -1]^T$, while class ω_2 (-1) consists of the vectors $\mathbf{x}_3 = [1, -1]^T$, $\mathbf{x}_4 = [1, 1]^T$.

- Draw** the points and make a conjecture about the line the (linear) SVM classifier will return.
- Using** the **dual representation of the SVM problem**, from ex. 1(c) derive
 - the **Lagrange multipliers** and
 - the **line** that separates the data from the two classes.
- Discuss** on the **results**.

Hints: 1. Defining $y_1=+1$, $y_2=+1$, $y_3=-1$, $y_4=-1$, substitute to the function

$$\left(\sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{ij} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right) \equiv J_1^*(\lambda)$$

y_i 's and \mathbf{x}_i 's and express $J_1^*(\lambda)$ only in terms of λ_i 's.

2. Taking the derivative of $J_1^*(\lambda)$ with respect to each λ_i and setting to zero, derive a system of equations for λ_i 's and find ALL its solutions.

3. Determine the $\boldsymbol{\theta}$ vector through $\boldsymbol{\theta} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i$

4. Determine the θ_0 parameter, using the constraints $\lambda_i [y_i (\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) - 1] = 0, i = 1, 2, \dots, N$

Exercise 2 (Python code + text):

Consider a two-class, two-dimensional classification problem for which you can find attached two **sets**: one for **training** and one for **testing** (file [HW9_ex2.mat](#)). Each of these sets consists of pairs of the form (y_i, \mathbf{x}_i) , where y_i is the **class label** for vector \mathbf{x}_i . Let N_{train} and N_{test} denote the number of training and test sets, respectively. The data are given via the following arrays/matrices:

- train_x** (a $N_{train} \times 2$ **matrix** that contains in its **rows** the **training** vectors \mathbf{x}_i)
- train_y** (a N_{train} -dim. column **vector** containing the **class labels** (0 or 1) of the corresponding **training** vectors \mathbf{x}_i included in **train_x**).
- test_x** (a $N_{test} \times 2$ **matrix** that contains in its **rows** the **test** vectors \mathbf{x}_i)

- test_y (a N_{test} -dim. column **vector** containing the **class labels** (0 or 1) of the corresponding **test** vectors \mathbf{x}_i included in test_x).

Train the **SVM classifier** using the training set given above and **measure** its **performance** using the test set, **using**: (a) the **linear kernel**, (b) the **polynomial kernel** and (c) **rbf kernel**. Perform **several runs** using the attached code, for **several choices of the parameters** included in each kernel and for **various values of C**.

Exercise 3:

Consider the following **binary** classification **two-class** problem (classes are labeled as 0 and 1)

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
y	1	0	0	0	0	0	0	1

where x_1, x_2, x_3 are the **input variables** and y the **class** where each triplet $(x_1 \ x_2 \ x_3)$ is assigned. **Prove** that this **classification problem** is **NOT linearly separable**.

Hint: Proceed using the **contradiction** method. Assume that there is a plane

(H): $\theta_1 x_1 + \dots + \theta_l x_l + \theta_0 = 0$

that separates the two triplets from class 1 from all the rest. This means that, for example, for the triplet (1,1,1) it is $\theta_1 + \dots + \theta_l + \theta_0 > 0$

Focus on (i) the triplets that belong to class 1 and (ii) the triplets that belong to class 0 and have only one coordinate equal to 1.

Exercise 4:

Consider the **lines** ($\epsilon 1$) $x_2=0$, ($\epsilon 2$) $x_1=0$ and ($\epsilon 3$) $x_1+x_2=2$ in the two-dimensional space that all **leave** the point (4,4) on their **positive side**. Consider a two-class classification problem where **class 1** contains all the points that lie on the positive side of all lines, as well as all the points that lie on the negative side of all lines. **Class 0** contains all points of the remaining (polygonal) regions

(i) Design the regions on the plane that correspond to each class.

(ii) Design a multilayer perceptron that solves the above classification problem, where each node is modeled by the relation $y = f(\mathbf{w}^T \mathbf{x} + w_0)$, where $f(z) = 1$, for $z > 0$ and

$f(z) = 0$, otherwise. Give the full architecture along with the weights and thresholds of each node (describe in some detail the steps you followed for designing the network).

Hint: (i) Use the point (4,4) to identify the positive and the negative sides of each line

(ii) Use the theory given in the lecture.

(iii) The equation of a plane that passes through the points (x_{11}, x_{12}, x_{13}) , (x_{21}, x_{22}, x_{23}) , (x_{31}, x_{32}, x_{33}) is

$$\begin{vmatrix} x_1 & x_2 & x_3 & 1 \\ x_{11} & x_{12} & x_{13} & 1 \\ x_{21} & x_{22} & x_{23} & 1 \\ x_{31} & x_{32} & x_{33} & 1 \end{vmatrix} = 0$$

Exercise 5:

Consider a two-class, two-dimensional classification problem for which you can find attached two **sets**: one for **training** and one for **testing** (file [HW9_ex5.mat](#)). Each of these sets consists of pairs of the form (y_i, x_i) , where y_i is the **class label** for vector x_i . Let N_{train} and N_{test} denote the number of training and test sets, respectively. The data are given via the following arrays/matrices:

- **train_x** (a $N_{train} \times 2$ **matrix** that contains in its **rows** the **training** vectors x_i)
- **train_y** (a N_{train} -dim. column **vector** containing the **class labels** (0 or 1) of the corresponding **training** vectors x_i included in **train_x**).
- **test_x** (a $N_{test} \times 2$ **matrix** that contains in its **rows** the **test** vectors x_i)
- **test_y** (a N_{test} -dim. column **vector** containing the **class labels** (0 or 1) of the corresponding **test** vectors x_i included in **test_x**).

Train a **neural network classifier** with a **single hidden layer** where the nodes have the **hyperbolic tangent output** function, for (a) 3 nodes, (b) 4 nodes, (c) 10 nodes, (d) 50 nodes (use the **MLPClassifier** Python **function** inserting properly the required parameters, see also the attached code), using the training set given above and **measure** the **performance** using the **test set**. Comment on the results.