“Machine Learning and Computational Statistics”

10th Homework

Exercise 1:

Wolfe dual representation: A convex programming problem is equivalent to

$$max_{\boldsymbol{\lambda} \geq \mathbf{0}} L(\boldsymbol{\theta}, \boldsymbol{\lambda})$$

$$\text{subject to } \frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbf{0}$$

Consider the SVM problem as it is stated in slide 6 of the 10th lecture. Prove that its equivalent dual representation is the one shown in slide 10.

Hints: (a) The parameters in SVM are $\boldsymbol{\theta}$ and $\theta_{\mathbf{0}}$. Using the Karush-Kuhn-Tacker (KKT) conditions (1) and (2), derive the equations given at the beginning of the 9th slide.

(b) Replace your findings to the Lagrangian function given in the 9th slide and perform operations.

(c) Use the Wolfe dual representation given above to state the dual form of the SVM problem (see also the 8th slide).

Exercise 2:

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

(a) **Draw** the points and make a conjecture about the line the (linear) SVM classifier will return.
(b) **Using** the dual representation of the SVM problem, from ex. 1(c) derive
   (i) the Lagrange multipliers and
   (ii) the line that separates the data from the two classes.
(c) **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 (keep in mind that the quantities $\boldsymbol{x}_i^T \boldsymbol{x}_j$ are scalars).

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, using the equations given in slide 10 of Lecture 10.
4. Determine the θ_0 parameter.

Exercise 3:

Consider the lines (ε1) $x_2=0$, (ε2) $x_1=0$ and (ε3) $x_1+x_2=2$ 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 $y = f(\boldsymbol{w}^T\boldsymbol{x} + w_0)$, 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 4 (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 *HW9a.mat*). Each of these sets consists of pairs of the form $(y_i, \boldsymbol{x}_i)$, where y_i is the class label for vector $\boldsymbol{x}_i$. 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} x2 **matrix** that contains in its rows the training vectors $\boldsymbol{x}_i$)

- ***train_y*** (a N_{train}–dim. column **vector** containing the **class labels** (0 or 1) of the corresponding training vectors $\boldsymbol{x}_i$ included in ***train_x***).
- ***test_x*** (a N_{test}x2 **matrix** that contains in its rows the test vectors $\boldsymbol{x}_i$)
- ***test_y*** (a N_{test}–dim. column **vector** containing the **class labels** (0 or 1) of the corresponding test vectors $\boldsymbol{x}_i$ 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 5 (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 *HW9b.mat*). Each of these sets consists of pairs of the form $(y_i, \boldsymbol{x}_i)$, where y_i is the class label for vector $\boldsymbol{x}_i$. 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} x2 **matrix** that contains in its rows the training vectors $\boldsymbol{x}_i$)
- ***train_y*** (a N_{train}–dim. column **vector** containing the **class labels** (0 or 1) of the corresponding training vectors $\boldsymbol{x}_i$ included in ***train_x***).
- ***test_x*** (a N_{test}x2 **matrix** that contains in its rows the test vectors $\boldsymbol{x}_i$)
- ***test_y*** (a N_{test}–dim. column **vector** containing the **class labels** (0 or 1) of the corresponding test vectors $\boldsymbol{x}_i$ 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.