OMML - Project 2

Professor: Laura Palagi
Team Members: Ahmed El Sheikh - 1873337
Amr Aly - 1848399
Katsiaryna Zavadskaya - 1847985

30 December 2019

Contents

1 Introduction

The aim of the project is to train the Non-Linear Support Vector Machine (SVM), namely to both set the values of the hyperparameters C and γ heuristically; and to find the values of the parameters α and b with an optimization procedure. For this last task we had to follow optimization procedures using 3 different algorithms:

1. *Full* Quadratic Programming (QP)
2. *Decomposed* Quadratic Programming
3. Most Violating Pair (MVP)

The data set which was used to build a classifier contains images that belong to different classes. Each sample is given as a 28x28 grayscale image together with its associated label. It was proposed to use Zalando's articles, which is a dataset of articles.
For Q1, Q2 and Q3 we used 1000 samples for each class: 2 and 4. In Q4 we added 1000 more images belonging to label 6. Scaling and shuffling of data was performed. In order to train and test models, the dataset was split into training and testing sets, with ratios 70% and 30%, respectively.

2 Full Quadratic Programming (QP)

The task of Question 1 is to find the solution of the SVM dual quadratic problem. In order to do that we used QP algorithm. Considering the convex quadratic dual problem of a soft margin SVM:

$$\begin{aligned} \min_{\alpha} \quad f = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{s.t.} \quad y^T\alpha = 0 \\ 0 \le \alpha \le Ce \end{aligned} \tag{1}$$

where P – is number of samples in the training set, C – a positive scalar, $\alpha \in \mathbb{R}^P$, $Q \in \mathbb{R}^{P\times P}$ with $Q_{ij} = y_i y_j K(x^i, x^j)$, $y_i \in \{-1, 1\}$, $e \in \mathbb{R}^P$ – a unit vector and $K(\cdot,\cdot)$ is a kernel function. We decided to use the RBF kernel, which defined as:

$$K(x, z) = e^{-\gamma\cdot\|x-z\|^2}$$

with $\gamma > 0$.
To find the solution of the SVM dual quadratic problem, the library CVXOPT was used, in particular cvxopt.solvers.qp.
Let α^* be the optimal solution of the QP. The corresponding nonlinear decision function is:

$$y(x) = sign\left(\sum_{p=1}^{P} \alpha_p^* y^p k(x^p, x) + b\right)$$

where b is a unique scalar for all the samples. b is computed using one of the values of vector α^*. In order to pick most secure value α_p^*, where $0 < \alpha_p^* < C$, we used the sample with the biggest non-zero α_p^*. Lets denote this sample $p = i$.
The optimal value b^* is determined from the complementarity conditions:

$$b^* = \frac{1 - y^i \cdot \left(\sum_{j=1}^{P} \alpha_j y^j k(x^j, x^i)\right)}{y^i}$$

In order to choose optimal values of hyperparameters, we used 5-fold cross-validation on the training set, varying C in range of $\{0.01, 0.1, 1, 2, 2.5, 3, 6, 10, 100\}$ and γ in $\{1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10\}$. The optimal values were found by assessing cross-validation accuracy. The highest reached cross-validation accuracy is 83.6% and was given by $C = 2.5$ and $\gamma = 0.01$.
The SVM with the RBF kernel trained using obtained hyperparameters has an accuracy of 100% on the training set and 84.5% on the test set. The model needs 27 seconds in order to perform 13 function evacuations, reaching the final value of the objective function of dual problem: -1896.62.

3 Decomposed Quadratic Programming

For all non-trivial datasets, the Q matrix described in (1) is large and dense to the extent that it can not be fully stored in memory. The Decomposed QP algorithm solves this complication by constructing and solving a series of smaller sub-problems, as shown in equation (2).

$$\begin{aligned} \min \ \frac{1}{2}\alpha_w^T \, Q_{ww} \, \alpha_w + \left(\alpha_{\bar{w}}^T \, Q_{\bar{w}w} \ - e_w^T\right)\alpha_w \\ \text{s.t.} \ \ y_w^T\alpha_w = -y_{\bar{w}}^T\alpha_{\bar{w}} \\ 0 \leq \alpha_W \leq C_w \end{aligned} \tag{2}$$

The sub-problems are constructed as follows, at each iteration, k, a working and non working sets W^k, $\bar{W}^k$ are constructed. α^k is accordingly partitioned into 2 sub-vectors $\left(\alpha_w^k, \alpha_{\bar{w}}^k\right)$. The sub-vector α_w^{k+1} is computed as the solution of equation (2). Similarly, the Q matrix is partitioned into the elements that are multiplied by α_w^k: Q_{ww}, the elements that are multiplied by $\alpha_{\bar{w}}^k$: $Q_{\bar{w}\bar{w}}$, and the elements multiplied by $\alpha_{\bar{w}}^k$: $Q_{w\bar{w}}$. The variables belonging to $\bar{W}$ remain unchanged. In order to define the Working set's selection rule, the following sets are defined:

$$R(\alpha) = L^+(\alpha) \cup U^-(\alpha) \cup \{i \mid 0 < \alpha_i < C\} \qquad S(\alpha) = L^-(\alpha) \cup U^+(\alpha) \cup \{i \mid 0 < \alpha_i < C\}$$

$$I(\alpha) = \left\{ i \mid i = \operatorname*{argmax}_{h \in R(\alpha)} - y_h(\nabla f(\alpha))_h \right\} \qquad J(\alpha) = \left\{ j \mid j = \operatorname*{argmin}_{h \in S(\alpha)} - y_h(\nabla f(\alpha))_h \right\}$$

where: L is the set of indices $\forall i$ where the corresponding α_i is equal to the lower bound: 0. L^+ includes those that have $y_i = 1$ and L^- includes those that have $y_i = -1$. Conversely, U is the set of indices $\forall i$ where the corresponding α_i is equal to the upper bound: C. U^+ includes those that have $y_i = 1$ and U^- includes those that have $y_i = -1$.
Given a desired **even** number, q, the algorithm selects $q_1 = \frac{q}{2}$ indices from $R\left(\alpha^k\right)$ such that:

$$-\frac{\nabla\mathcal{F}(\alpha^k)_{i^1(k)}}{y_{i^1(k)}} \geq -\frac{\nabla\mathcal{F}(\alpha^k)_{i^2(k)}}{y_{i^2(k)}} \geq \cdots \geq -\frac{\nabla\mathcal{F}(\alpha^k)_{i^{q_1}(k)}}{y_{i^{q_1}(k)}}$$

and $q_2 = \frac{q}{2}$ indices from $S\left(\alpha^k\right)$ such that:

$$-\frac{\nabla\mathcal{F}(\alpha^k)_{j^1(k)}}{y_{j^1(k)}} \leq -\frac{\nabla\mathcal{F}(\alpha^k)_{j^2(k)}}{y_{j^2(k)}} \leq \cdots \leq -\frac{\nabla\mathcal{F}(\alpha^k)_{j^{q_2}(k)}}{y_{j^{q_2}(k)}}$$

where: $i^1(k) \in I(\alpha^k)$ and $j^1(k) \in J(\alpha^k)$. W^k is defined as $W^k = \{i^1, ..., i^{q_1}, j^1, ..., j^{q_2}\}$.
Since q_1, q_2 are selected based on $\frac{\nabla f(\alpha^k)}{y}$, after solving equation (2) to obtain α_w^{k+1}, The gradient is updated according to the following rule:

$$\nabla f(\alpha^{k+1}) = f(\alpha^k) + \sum_{i \in W^k} Q_i\left(\alpha^{k+1} - \alpha^k\right)$$

In order to be able to construct the new working and non working sets W^{k+1}, $\bar{W}^{k+1}$. The algorithm keeps running until the KKT condition is satisfied, namely, when the difference between $m(\alpha^k)$ and $M(\alpha^k)$ is $\leq 10^{-3}$. And when no numerical precision problems are present this becomes:

$$m(\alpha^k) = \max_{i \in R(\alpha^k)} \{-y_i(\nabla f(\alpha^k))_i\} \leq \min_{j \in S(\alpha^k)} \{-y_j(\nabla f(\alpha^k))_j\} = M(\alpha^k)$$

$$d_h^{ij} = \begin{cases} y_i & \text{if } h = i \\ -y_j & \text{if } h = j \\ 0 & \text{otherwise} \end{cases}$$

After Running the algorithm, taking 70% of the dataset as training data, and 30% as testing data, the algorithm took 0.185 seconds (total of 14 iterations) to finish training achieving results of 80% for acc on training set and 94.0% for acc on testing set.

4 Most Violating Pair (MVP)

The MVP algorithm is an instance of Decomposed QP where the q is strictly equal to 2 (q = 2). Consequently, $W^k = \{i^k, j^k\}$ and Q_{ww} is a 2x2 matrix.
For each iteration, the analytic solution of the sub-problem provided below was used.

$$\min\ f(\alpha) = \frac{1}{2}\alpha^T \begin{pmatrix} q_{ii} & q_{ij} \\ q_{ji} & q_{jj} \end{pmatrix} \alpha - \alpha_i - \alpha_j$$

$$t.c. \qquad y_i\alpha_i + y_j\alpha_j = b$$

$$0 \leq \alpha_h \leq C \qquad h = i, j,$$

The algorithm terminates when the KKT condition, described above, is reached. Alpha is updated (at each iteration) according to the following rule:

$$\alpha^{k+1} = \alpha^k + td$$

where:

$$t = \frac{-\nabla f(\alpha^k)^T d^k}{d^{kT} Q d^k}$$

$$d_h^{ij} = \begin{cases} y_i & \text{if } h = i \\ -y_j & \text{if } h = j \\ 0 & \text{otherwise} \end{cases}$$

After Running the algorithm, taking same number of samples as Decomposition algorithm, the algorithm took 0.6 seconds (total of 693 iterations) to finish training achieving results of 93.93% for accuracy on training set and 80.17% for accuracy on testing set.

5 Multi-class SVM

A multi-class SVM was implemented to classify between classes 2, 4 and 6 (provided in the given dataset). The One Against All method was preferred to the One Against One method, as it achieved better results despite the unbalanced dataset problem. 3 classifiers were used, comparing each class to all the others with all the other classes combined to form the opposing class i.e. [class 2 against classes 4 and 6]. The number of iterations was set to 100 iterations to train the 3 classifiers, in 21.83 seconds. An accuracy of 100% was achieved on the training sets, proving that the Nonlinear SVM performs extremely well on the dataset. After the 3 classifiers were trained, the final prediction was given based on the projection of a point onto each classifier, which was assigned a class label with the maximum projection. For this question only the polynomial kernel was used.

6 Results

SVM Hyperparams used:

- C = 2.5
- $\gamma = 0.01$

Exercise	Train Acc (%)	Test Acc (%)	Opt Time (sec)	Iterations	Final Obj Fn
1	100	84.5	24.8	7	-494.95
2	94.00	80.0	0.185	14	-3464.999
3	93.93	80.17	0.56	693	-3465.00

Table 1: Results for Questions 1,2 and 3