Project Report

Stepic (1861402), Nurasilova (1848425), Nastasic (1852026)

OMML - Project II

December 30, 2019

Contents

Introduction

The goal of the project is to build a classifier that distinguishes between images of Zalando's articles that belong to different classes. In the first three questions binary classifier is implemented, recognizing images of pullovers and coats identified by the labels 2 and 4, respectively. For the question four, three-class classifier which recognizes images of pullovers, coats and skirts is made with labels 2, 4 and 6, respectively. Data points are 28x28 grayscale images, associated with a label.

We worked with a non-linear SVM and a non-linear decision function:

$$y(x) = sign(\sum_{p=1}^{P} \alpha_p y^p K(x^p, x) \ + \ b)$$

where $K(x^p, x)$ is the Gaussian Kernel $K(x, y) = e^{-\gamma||x-y||^2}$. The Gaussian Kernel choice is motivated by the fact that this Kernel is generally more flexible than the polynomial, even though the polynomial is computationally less demanding.

Q1: SVM dual quadratic problem

We implemented **Grid Search** to find the best combination of hyper-parameters C and γ. All the combinations of C from values [0.1,0.5,0.7,0.8,0.9,1.1,2.1] and γ from [0.1,0.01,0.001,0.0001] are tested.

Firstly, the data was split into training and test data set. Training set was further split into training and validation data set. **Misclassification error**, percentage of wrongly classified points was a measure of goodness of classification. **10-Fold Cross-Validation** technique on optimization routine with misclassification error as an objective function was used in order to prevent overfitting, to simulate behaviour on unseen (validation) data. The pair of C and γ that gave the smallest average misclassification error on validation data set was chosen.

The final setting of hyper-parameters is $\mathbf{C = 2.1}$ and $\boldsymbol{\gamma} = \mathbf{0.001}$.

To solve the quadratic minimization problem we used the **solvers.qp** from the **CVXOpt** library with default **OSQP** solver for quadratic problems with **absolute accuracy** and **feasible tollerance** set up to **1e-12**. From the result, we extracted vector α, and identify the Support Vectors, which correspond to $\boldsymbol{\alpha}$ values greater than a given threshold. We fixed a threshold equal to $\boldsymbol{e^{-5}}$ below which all the values are considered to be 0. Identified SV are used for prediction on training and test data set.

Procedure finishes with the following results:

Number of function evaluations:	**11**
Starting objective function value:	**0**
Ending objective function value:	**-848.6648**
Seconds:	**2.0934**
KKT violation:	**3.2733e-06**

In terms of accuracy results are following:

Training accuracy	Validation accuracy	Test accuracy
0.9531	0.8781	0.8775

Confusion matrices:

Training	**True label 2**	**True label 4**
Predicted label 2	**772**	**29**
Predicted label 4	**46**	**753**

Test	**True label 2**	**True label 4**
Predicted label 2	**162**	**29**
Predicted label 4	**20**	**189**

Q2: SVM decomposition method

In this section, we used the hyper-parameters that we found as optimal in previous section. They were $\mathbf{C = 2.1}$ and $\boldsymbol{\gamma = 0.001}$. And we used the same kernel: Gaussian. The values of the $\boldsymbol{q = 10}$. We implemented a procedure which finds working set of indexes in each iteration according to certain rule. In the rule we consider two sets R and S. We take indexes from each set in a way where a direction will be feasible. And if function can not find it we use $\boldsymbol{SVM^{light}}$ strategy. Where we sort R in descending and S in ascending order. Then we are taking first 5 elements from each. Then we repeat this procedure until difference between m and M will be less than 0.00001, where $m = max(\frac{-\nabla f(\alpha_i)}{y_i})$ and $M = min(\frac{-\nabla f(\alpha_j)}{y_j})$.

To solve the quadratic minimization problem we used the **solvers.qp** from the **CVXOpt** library with default **OSQP** solver for quadratic problems with **absolute accuracy** and **feasible tollerance** set up to **1e-12**.

We obtained the following results by execution implemented procedure:

Training accuracy:	**0.86**
Test accuracy:	**0.83**
Number of iterations:	**338**
Number of function evaluations:	**1971**
Starting objective function value:	**0**
Ending objective function value:	**7.6249e-07**
Seconds:	**116**
KKT violation:	**5.5286e-07**

Confusion matrices:

Training	**True label 2**	**True label 4**
Predicted label 2	**753**	**29**
Predicted label 4	**195**	**623**

Test	**True label 2**	**True label 4**
Predicted label 2	**203**	**15**
Predicted label 4	**52**	**130**

Q3: The most violating pair (MVP) decomposition method

In this point we used the same hyper-parameter configuration of the previous part, $C = 2.1$ and $\gamma = 0.001$ and we fixed $\boldsymbol{q = 2}$. This is the special case SVM^{light} algorithm, called Maximum Violating Pair because among all the pairs that violate KKT conditions, it is using the pair which violates it the most. This approach defines, among all feasible direction one of the steepest descent.

We implemented the analytical solution, to solve the sub problems. This approach guarantees to find the maximum feasible step size t along the direction d starting from $\alpha_0 = 0$. First, we found the optimal direction d^* and a feasible value for t then, we evaluated the maximal possible step t_{max}. After that, the optimal step size t^* is found by $t^* = min\{t, t_{max}\}$. Finally, we updated the Lagrangian multipliers α^*.

Once we have the optimal α^* values, we iterate until one of these stop criteria happens:

- KKt condition satisfied: $max(\frac{-\nabla f(\alpha_i)}{y_i}) - min(\frac{-\nabla f(\alpha_j)}{y_j}) \leq 0.00001$
- Maximal number of 10000 iterations reached

Procedure finishes with the following results:

Training accuracy:	**0.9531**
Test accuracy:	**0.8775**
Number of iterations:	**4600**
Starting objective function value:	**0**
Ending objective function value:	**-848.665**
Seconds:	**42.42**
KKT violation:	**9.739e-06**

In terms of accuracy results are following:

Training accuracy	**Validation accuracy**	**Test accuracy**
0.9531	**0.8781**	**0.8775**

Obtained results in terms of accuracy are the same like in Q1. Confusion matrices are also the same as in Q1.

Q4: Multi-class classification

There are two strategies that could be used in dealing with multi-class classification. One is One-vs-All, while the other one is One-vs-One. Since we have the same sample size for each of the classes, it is better to train the model using the One-vs-One approach. This strategy practically means training three classifiers, whereby classifiers are trained to make a difference between each pair of classes. In other words, training three classifiers to distinguish between label2 VS label4, label2 VS label6, and label4 VS label6, respectively.

After making predictions for each of the classifiers, we combined the results by taking the most frequent prediction for each sample.

We used the hyper-parameter configuration discovered as the optimal one at the beginning of the analysis. In other words, the values of the hyper-parameters used are C = 2.1 and = 0.001, and the Gaussian kernel.

The results for each of the classifiers, along with the hyper-parameters used, are shown in the table below.

Classifier	C	γ	tr. acc.	test acc.	fun evals	Init obj. fun.	Final obj. fun.	KKT viol	CPU time
Cl. 2 4	2.1	0.001	0.95	0.88	9	0	-880.44	0.025	2.01
Cl. 2 6	2.1	0.001	0.96	0.91	10	0	-863.18	0.007	2.46
Cl. 4 6	2.1	0.001	0.96	0.9	10	0	-770.54	0.003	2.28

Confusion matrix for test data (all classifiers):

Training	**True label 2**	**True label 4**	**True label 6**
Predicted label 2	**158**	**13**	**20**
Predicted label 4	**27**	**166**	**14**
Predicted label 6	**13**	**17**	**172**

Procedure finishes with the following results:

Training accuracy:	**0.92**
Test accuracy:	**0.83**
Number of function evaluations:	**29**
KKT violation:	**0.012**
Computational time (seconds):	**8.97**

Summary

Summary of all the results is represented in the **Table 1**.

Table 1.

Q	C	γ	q	tr. acc.	test acc.	number its	number fun evals	KKT viol	CPU time
Q1	2.1	0.001	X	0.9531	0.8775	/	11	3.27e-06	2.09
Q2	2.1	0.001	10	0.86	0.83	338	1971	5.53e-07	116
Q3	2.1	0.001	2	0.9531	0.8775	4600	/	9.74e-06	42.42
Q4	2.1	0.001	X	0.92	0.83	/	29	0.012	8.97

Table 1: Final table of results