OTDM Constrained Optimization Lab Assignment: Support Vector Machines in AMPL

Mariem Gandouz, Egon Ferri

21th of NOVEMBER 2019

Contents

1 Introduction

This report presents the Support Vector Machine (SVM) learning algorithm. We will need to first talk about margins and the idea of separating data with a large gap. Next, we will talk about the optimal margin classifier, which will lead us to talk about the duality for the SVM. We will also implement SVMs in AMPL on random generated data with primal and dual formulations, and finally, we will close off our report with a classic iris dataset example analyse.

2 Modelling SVMs

2.1 Non-linear separable data, Soft-Margin SVM

In general, the training data can rarely be separated linearly because they are often noisy. An outlier in the training data, caused by noise, for example an observation with a wrong class label, could clearly influence the position of the hyperplane. To avoid this problem, it is possible to use a slightly modified variant of the Hard-Margin SVM, that can tolerate a certain number of outliers. The goal is to find a hyperplane that allows some misclassifications. For this purpose, the constraint in the linearly separable case $y_i(w^\top x_i + \gamma) \geq 1(*)$[1] is eased, that's why the resulting SVM is referred as Soft-Margin SVM. Furthermore, the slacks $s_i \geq 0$ are inserted for each observation x_i. Each slack indicates the deviation of an observation x_i from the separating hyperplane and can be regarded as a penalty for this observation if x_i does not satisfy the previous constraint (*). Specifically, these are given by:

1) $s_i = 0$, for correctly classified observations, i.e. if

$$y_i(w^\top x_i + \gamma) \geq 1$$

2) $0 < s_i \leq 1$, for observations within the separation area of the hyperplane that are on the correct side of the hyperplane,i.e. if

$$0 \leq y_i(w^\top x_i + \gamma) < 1$$

3) $s_i > 1$, for misclassified observations, i.e. if the observations are on the wrong side of the hyperplane and thus

$$y_i(w^\top x_i + \gamma) < 0$$

Based on the above considerations, the constraints of the modified formular result as follows:

$$y_i(w^\top x_i + \gamma) \geq 1 - s_i, s_i \geq 0 \text{ for } i = 1, \dots, m$$

[1]Given m pairs $(x_i, y_i) \in \mathbb{R}^n \times \{+1, -1\}$ and the hyperplane defined by $(w, \gamma) \in \mathbb{R}^{n+1}, i = 1, \dots, m$

Thus the total deviation error is given by $\sum_{i=1}^{m} s_i$. Since the observations are not classified correctly with $s_i > 1$, so $\sum_{i=1}^{m} s_i$ is an upper bound for the number of misclassified observations. A hyperparameter $\nu > 0$ is introduced to regulate the influence of the slacks. This hyperparameter determines the tradeoff between two opposing goals, namely minimizing the number of missclassified points and maximizing the width of the margin. In other words the aim of ν is to achieve the best ratio between the widest possible margin and the lowest possible number of misclassifications. Accordingly, the optimization problem

$$\begin{array}{lll} \min_{(w,\gamma)\in\mathbb{R}^{n+1}} & \frac{1}{2}w^\top w & \\ \text{s. to} & y_i(w^\top x_i + \gamma) \geq 1 & i = 1, \ldots, m \end{array}$$

is adjusted as follows:

$$\begin{array}{l} \min\limits_{(w,\gamma,s)\in\mathbb{R}^{n+1+m}} \dfrac{1}{2}w^\top w + \nu \sum\limits_{i=1}^{m} s_i \\ \text{s. to } y_i\left(w^\top x_i + \gamma\right) + s_i \geq 1 \quad i = 1, \ldots, m \\ \qquad s_i \geq 0 \end{array} \quad i = 1, \ldots, m \qquad (**)$$

Now, all that remains is to talk about the duality for SVM.

2.2 Dual problem formulation

Similar to the procedure in the last section of our report, the SVM optimization problem is solved in the form of the dual problem instead of the primary problem (**).
It can be thus shown that, the dual problem can be formulated as follows:

$$\begin{array}{ll} \max & \sum_{i=1}^{m} \lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i\lambda_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \\ \text{s. to} & \sum_{i=1}^{m} \lambda_i y_i = 0 \\ & 0 \leq \lambda_i \leq \nu \quad i = 1, \ldots, m \end{array}$$

The dual problem is a quadratic convex optimization problem. The slacks and the Lagrange multipliers μ_i are not present in this dual problem.

Based on KKT conditions the point w^* can be determined equivalent to the linearly separable case and is given by

$$w^* = \sum_{i=1}^{m} \lambda_i^* y_i x_i \qquad (1)$$

For the determination of the parameter γ^* we consider any support vector x_s (with the class label y_s) with $0 < \lambda_s < \nu$. Thus

$$y_s\left(\langle \mathbf{w}^*, \mathbf{x}_s\rangle + \gamma^*\right) = 1$$

Then we compute γ^* as

$$\gamma^* = \tfrac{1}{y_s} - \langle \mathbf{w}^*, \mathbf{x}_s\rangle \qquad y_s = \pm 1 \tag{2}$$

Let's move on. In the next section, we will exploit these ideas to our classification problem.

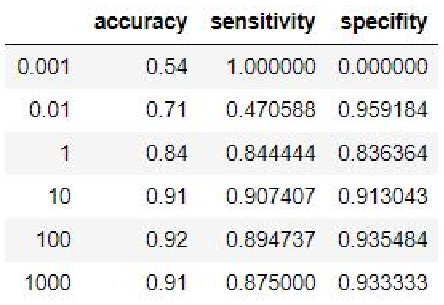
3 Generated data

3.1 Our generator

Since we had several problems with the given generator, we decide to write one of our own in Python, and a function that, taken the random generated data, is capable to spit out data in a format ready for AMPL. The function that we used can be found in the functions.py python script. The concept of the generator is the same of the given one: It randomly generates points n points $x \in \mathbb{R}^4$: if $\sum_{i=1}^4 \geq 2$ the point belongs to class $+1$; otherwise, to class -1. We added also a probability of 0.05 to misclassificate, in order to have not-linearly separable data.

3.2 The choice of ν

$\nu \geq 0$ is a parameter that we have to fix a priori.
The goal is now to find the best value for ν, so we tried different ν's and these are the results:



	accuracy	sensitivity	specifity
0.001	0.54	1.000000	0.000000
0.01	0.71	0.470588	0.959184
1	0.84	0.844444	0.836364
10	0.91	0.907407	0.913043
100	0.92	0.894737	0.935484
1000	0.91	0.875000	0.933333

Larger ν means we assign more importance to reducing number of misclassified points, on the other hand, smaller ν means we assign more importance to having a large margin.
We discovered that with small ν results are really bad in terms of accuracy, but once we arrive to reasonably large , the accuracy is stable around the 90%, with a little bit of noise due to the randomicity of the training and the testing data. From now on, we will consider as a given ν of 10.

3.3 Results obtained with SVM

In the primal formulation we obtained:

$w^* = [4.54007, 5.06347, 5.98317, 5.45954]$

$\gamma^* = -10.2623$

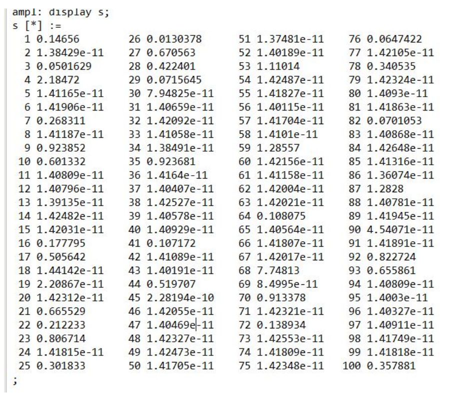
3.4 Primal Dual Relationship

We want to show, that for the dual, we have the same hyperplane as the primal.
From (1) and (2) let's compute w^* and γ^*:

For w^*, we could do this directly in AMPL with a "fake constraint":

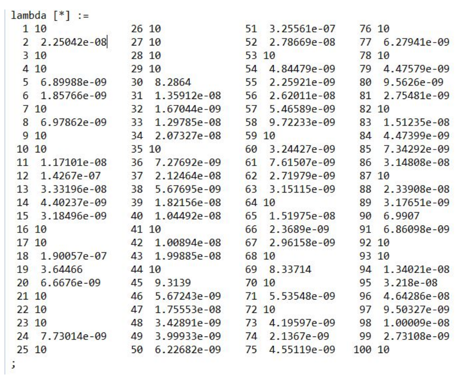
```
subject to get-w{i in 1..m}:
     w[i]=sum{j in 1..n}lambda[j]*y[j]*x[j,i];
```

Instead for γ^*, we need at first to identify a point displaying λ and *slacks* in AMPL.



```
ampl: display s;
s [*] :=
  1 0.14656         26 0.0130378      51 1.37481e-11    76 0.0647422
  2 1.38429e-11     27 0.670563       52 1.40189e-11    77 1.42105e-11
  3 0.0501629       28 0.422401       53 1.11014        78 0.340535
  4 2.18472         29 0.0715645      54 1.42487e-11    79 1.42324e-11
  5 1.41165e-11     30 7.94825e-11    55 1.41827e-11    80 1.4093e-11
  6 1.41906e-11     31 1.40659e-11    56 1.40115e-11    81 1.41863e-11
  7 0.268311        32 1.42092e-11    57 1.41704e-11    82 0.0701053
  8 1.41187e-11     33 1.41058e-11    58 1.4101e-11     83 1.40868e-11
  9 0.923852        34 1.38491e-11    59 1.28557        84 1.42648e-11
 10 0.601332        35 0.923681       60 1.42156e-11    85 1.41316e-11
 11 1.40809e-11     36 1.4164e-11     61 1.41158e-11    86 1.36074e-11
 12 1.40796e-11     37 1.40407e-11    62 1.42004e-11    87 1.2828
 13 1.39135e-11     38 1.42527e-11    63 1.42021e-11    88 1.40781e-11
 14 1.42482e-11     39 1.40578e-11    64 0.108075       89 1.41945e-11
 15 1.42031e-11     40 1.40929e-11    65 1.40564e-11    90 4.54071e-11
 16 0.177795        41 0.107172       66 1.41807e-11    91 1.41891e-11
 17 0.505642        42 1.41089e-11    67 1.42017e-11    92 0.822724
 18 1.44142e-11     43 1.40191e-11    68 7.74813        93 0.655861
 19 2.20867e-11     44 0.519707       69 8.4995e-11     94 1.40809e-11
 20 1.42312e-11     45 2.28194e-10    70 0.913378       95 1.4003e-11
 21 0.665529        46 1.42055e-11    71 1.42321e-11    96 1.40327e-11
 22 0.212233        47 1.40469e-11    72 0.138934       97 1.40911e-11
 23 0.806714        48 1.42327e-11    73 1.42553e-11    98 1.41749e-11
 24 1.41815e-11     49 1.42473e-11    74 1.41809e-11    99 1.41818e-11
 25 0.301833        50 1.41705e-11    75 1.42348e-11   100 0.357881
;
```

Figure 1: Slacks



```
lambda [*] :=
  1 10              26 10              51  3.25561e-07    76 10
  2  2.25042e-08    27 10              52  2.78669e-08    77  6.27941e-09
  3 10              28 10              53 10              78 10
  4 10              29 10              54  4.84479e-09    79  4.47579e-09
  5  6.89988e-09    30  8.2864         55  2.25921e-09    80  9.5626e-09
  6  1.85766e-09    31  1.35912e-08    56  2.62011e-08    81  2.75481e-09
  7 10              32  1.67044e-09    57  5.46589e-09    82 10
  8  6.97862e-09    33  1.29785e-08    58  9.72233e-09    83  1.51235e-08
  9 10              34  2.07327e-08    59 10              84  4.47399e-09
 10 10              35 10              60  3.24427e-09    85  7.34292e-09
 11  1.17101e-08    36  7.27692e-09    61  7.61507e-09    86  3.14808e-08
 12  1.4267e-07     37  2.12464e-08    62  2.71979e-09    87 10
 13  3.33196e-08    38  5.67695e-09    63  3.15115e-09    88  2.33908e-08
 14  4.40237e-09    39  1.82156e-08    64 10              89  3.17651e-09
 15  3.18496e-09    40  1.04492e-08    65  1.51975e-08    90  6.9907
 16 10              41 10              66  2.3689e-09     91  6.86098e-09
 17 10              42  1.00894e-08    67  2.96158e-09    92 10
 18  1.90057e-07    43  1.99885e-08    68 10              93 10
 19  3.64466        44 10              69  8.33714        94  1.34021e-08
 20  6.6676e-09     45  9.3139         70 10              95  3.218e-08
 21 10              46  5.67243e-09    71  5.53548e-09    96  4.64286e-08
 22 10              47  1.75553e-08    72 10              97  9.50327e-09
 23 10              48  3.42891e-09    73  4.19597e-09    98  1.00009e-08
 24  7.73014e-09    49  3.99933e-09    74  2.1367e-09     99  2.73108e-09
 25 10              50  6.22682e-09    75  4.55119e-09   100 10
;
```

Figure 2: Lambdas

We see that the point 19 is the first one that respects our constraints. We can use this point to retrive γ^*, with the help of python:

```
gamma_retrieved=1 - np.matmul(W,x_19)
```

that again returns our beloved γ^*:

$\gamma^* = -10.2623$

4 Real data

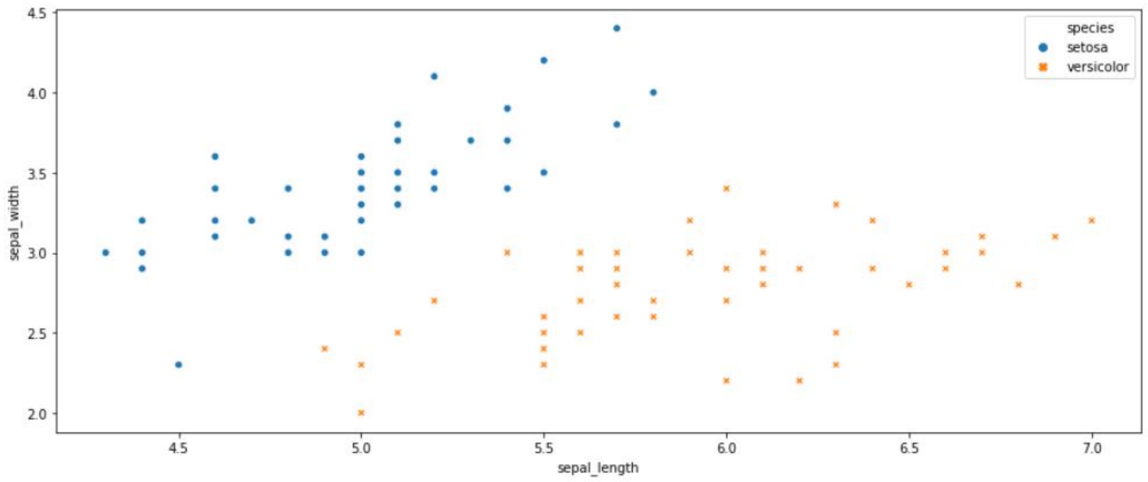
Figure 3: Iris setosa

Figure 4: Iris versicolor

4.1 The iris dataset

To test our SVM we decided to use a simple and famous dataset; the Iris dataset. Since we wanted only two classes (to feed the SVM) and two features (to have the possibility to have an easy and immediate graphical representation), we filtered the data set in order to have only two species; setosa and versicolor, and two attributes; the length and the width of the sepal.

Figure 5: Filtred iris dataset



We split our data into two subsets: training data (taking at random 80% of the data) and testing data (the rest).

Figure 6: Testing data

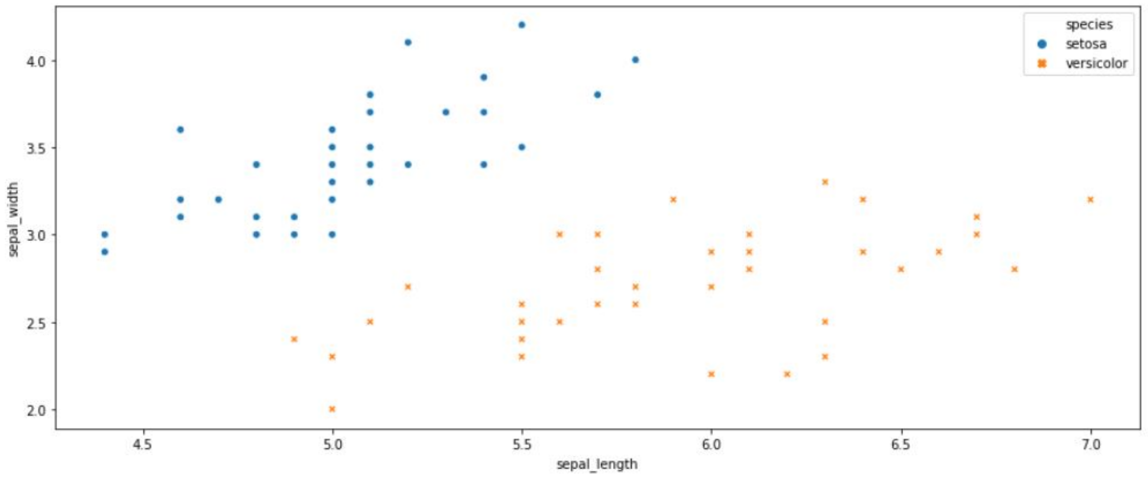
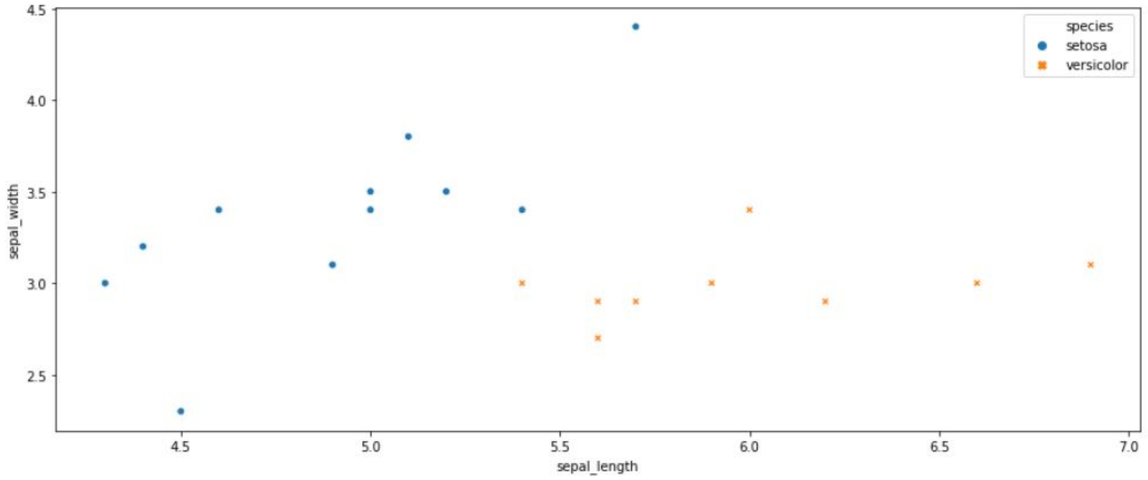


Figure 7: Training data



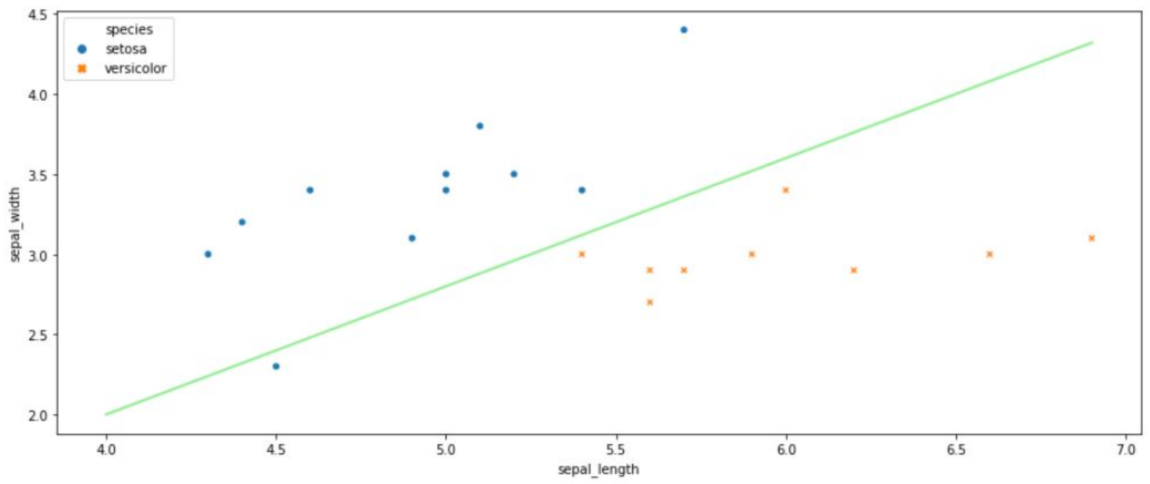
4.2 Results obtained with SVM

We feed the training data to our AMPL SVM, and we used the parameters w_1, w_2, γ to recollect the optimal margin classifier with the equation:

$$y = -(\frac{w_1}{w_2}x + \frac{\gamma}{w_2})$$

applying what we found training our model to the test data, we found these results:

Figure 8: test classification



Our SVM seems to work very well. It only fails on one point, but taking a glance at our test (or full) dataset we see that our single wrong classified point is in some sense an outlier of his class, so its misclassification could be expected.

5 Conclusion

In this project, we have seen how to implement a SVM with primal and dual formulations in AMPL that is going to be trained to solve our classification problem. We also have tested the influence of the hyperparameter ν and how to come up with a way of efficiently finding this hyperparameter that minimize the mistakes and keep the margin as wide as possible. We found that the best accuracy is obtained with $\nu = 10$. Setting this hyperparameter is very important to train our SVM. Since SVM is a convex optimisation problem, we found as expected that the dual problem has the same optimum solution as our original, primal problem.

6 Resources

- AMPL model:

  svm_basic.mod $--->$ Primal formulation of SVM

  svm_dual.mod $--->$ Dual formulation of SVM

- Datasets:

  train.dat $--->$ The main training data that we used ($\nu = 10$)

  train_001.dat $--->$ Dataset with $\nu = 0.01$

  train_01.dat $--->$ Dataset with $\nu = 0.1$

  train_1.dat $--->$ Dataset with $\nu = 1$

  train_100.dat $--->$ Dataset with $\nu = 100$

  train_1000.dat $--->$ Dataset with $\nu = 1000$

  iris.csv $--->$ Whole iris dataset

  iris_train_ampl.dat $--->$ iris dataset modified to feed AMPL

  train_iris $--->$ Trainset of iris

  test_iris $--->$ Testset of iris

- Python codes:

  functions.py $--->$ Function that we used in the main notebook stored here for tidiness

  SVM.ipynb $--->$ Main notebook that contains the Python part of the analysis