Support Vector Machines in AMPL

Course Assignment II - Optimization Techniques for Data Mining

Alex Martorell
Pim Schoolkate

Facultat d'Informàtica de Barcelona

2021
Universitat Politècnica de Catalunya

Contents

1 Introduction

Support Vector Machines (SVM) are powerful classification algorithms that make use of kernels and hyper-planes to separate data. In this report, two different SVMs making use of a linear kernel, are showcased both in their primal and dual formulation. The first SVM was trained on quasi linear separable data that contains misclassifications, separating two ambiguous classes -1 and 1. This data was generated using the provided `gensvmdat-mac` file, giving `seed=1`. The second SVM was trained to differentiate between the handwritten digits 0 and 1. For both SVMs the weights w, gamma γ, total errors and accuracy are reported, and since these are the same for the primal and dual formulation they are only presented once.

2 Method

Both the primal and dual formulation of the SVMs were programmed in `AMPL`, and are discussed separately below.

2.1 Primal formulation

The primal formulation of a SVM revolves around minimizing the following objective function:

$$\min_{(w,\gamma,s)\in\mathbb{R}^{n+1+m}} \frac{1}{2}W^TW + \nu\sum_{i=1}^{m} s_i$$

Subject to the constraints:

$$y_i(W^Tx_i + \gamma) + s_i \geq 1, \quad i = 1,...,\mathrm{m}$$

$$s_i \geq 0, \quad i = 1,...,\text{m}$$

Here, w are the weights and γ the intercept of the hyper-plane $w^T x + \gamma = 0$. s_i are the slacks, one defined for each point, equalling 0 if the point is a support vector or falls on the right side of the hyper-plane, and > 0 if the point is a misclassified point. Lastly, ν is a hyper-parameter giving weight to the misclassified points.

The implementation of the primal was done as follows in AMPL:

```
param m; param n; param nu;

param y {1..m}; param A {1..m, 1..n};

var W {1..n}; var g; var s {1..m} >= 0;

minimize fobj:
   sum {i in 1..n} 1 / 2 * W[i] * W[i] + nu * sum {j in 1..m} s[j];

subject to Soft_constraints {j in 1..m}:
     y[j] * ( sum {i in 1..n} (W[i]*A[j, i]) + g) + s[j] >= 1;
```

The complete run script for computing the primal, its errors and accuracy was implemented as follows:

```
reset;
model primal.mod;
data train_data.dat;
option solver cplex;
solve;
display W, g;

reset data nu, m, n, A, y;
data test_data.dat;
param y_pred {1..m};

let {i in {1..m}} y_pred[i] := g + sum{j in {1..n}} W[j]*A[i,j];
let {i in {1..m}} y_pred[i] := if y_pred[i] <= 0 then -1 else 1;

param errors; let errors := 0;

for {i in {1..m}} {
   if y_pred[i] != y[i] then
      let errors := errors + 1;
}

param acc = (m - errors) / m;

display errors, acc;
```

2.2 Dual formulation

The dual formulation of the SVM aims to maximize the Lagragian $L(w, \gamma, s, \lambda, \mu)$, λ and μ being the Lagragian multipliers of the Karush-Kuhn-Tucker conditions of the SVM primal formulation. The objective function of the dual formulation has been shown to be:

$$\max_{\lambda} \lambda^T e - \frac{1}{2}\lambda^T Y A A^T Y \lambda$$

Subject to the constraints:

$$\lambda^T Y e = 0$$

$$0 \leq \lambda \leq \nu$$

Here, λ is the value to be optimized, A is a matrix of kernel transformed values of x such that $A[i] = \phi(x_i)$, Y is a diagonal matrix containing the predictor values y_i and e is an $1 \times m$ vector of ones.

To compare the results of the dual formulation with the primal formulation, the weights w can be computed using the optimal values for λ, namely:

$$W = \sum_{i=1}^{m} \lambda_i y_i \phi(x_i)$$

The true value for γ can also be found by finding a support vector, satisfying $s_i = 0$ and $\nu > \lambda_i > 0$

$$\gamma = \frac{1}{y_i} - W^T \phi(x_i)$$

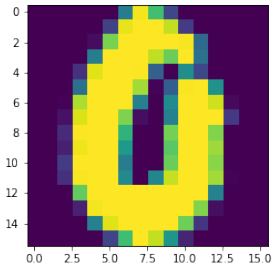
The complete dual formulation was implemented in AMPL as follows. Note that the constraint "Weights" does not actually impose a constraint, but rather directly helps to compute the weights for the dual, making it easier to compare it with the primal formulation.

```
param m; param n; param nu;

param y {1..m}; param A {1..m, 1..n};

var w {1..n}; var l {1..m} >= 0, <= nu;

maximize fojb:
   sum {i in 1..m} l[i] - 1/2 * sum {j in 1..m, i in 1..m, p in
       1..n} l[i] * y[i] * l[j] * y[j] * A[i, p] * A[j, p];

subject to Constraints:
    sum {j in 1..m} y[j] * l[j] = 0;
```

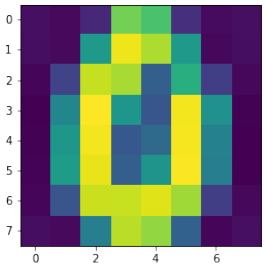
```
subject to Weights {i in 1..n}:
   w[i] = sum {j in 1..m} l[j] * A[j, i] * y[j];
```

In order to calculate the value for γ, the amount of errors and the accuracy, the following run scripts was created:

```
reset;
model dual.mod;
data train_data.dat;
option solver cplex;
solve;

param count; param gamma; param gamma_sum;
let gamma_sum := 0; let count := 0;

for {i in 1..m}{
   if nu-0.025 > l[i] > 1e-3 then {
      let gamma_sum := gamma_sum + 1/y[i] - sum{j in {1..n}}
          w[j]*A[i,j];
      let count := count + 1
   }
}

let gamma := gamma_sum / count;

display w, gamma;

reset data nu, m, n, A, y;
data test_data.dat;
param y_pred {1..m};

let {i in {1..m}} y_pred[i] := gamma + sum{j in {1..n}} w[j]*A[i,j];
let {i in {1..m}} y_pred[i] := if y_pred[i] <= 0 then -1 else 1;

param errors; let errors := 0;

for {i in {1..m}} {
   if y_pred[i] != y[i] then
      let errors := errors + 1;
}

param acc = (m - errors) / m;
display errors, acc;
```

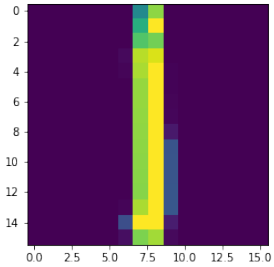


(a) 0 of size 64×64

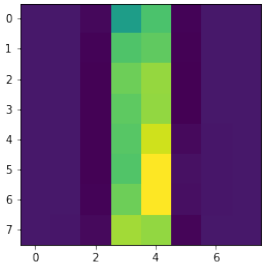


(b) 0 of size 8×8

Figure 2.1: Graphical representation of the data of class -1



(a) 1 of size 64×64



(b) 1 of size 8×8

Figure 2.2: Graphical representation of the data of class 1

3 Data

3.1 Generated data

As explained above, one SVM was trained on generated linearly separable data using the `gensvmdat-mac` script with `seed=1`. For the train set, 800 data points were used, whereas the test set contained 200 data points.

3.2 Handwritten digits

Next to this, a SVM was trained on a dataset of the handwritten digits 0 (coded as -1) and 1 (coded as 1). The original images were of size 16×16 making up for a total of 256 variables. Unfortunately, and quite counter intuitively (as the dual is supposed to be more effective for large amount of columns), the dual formulation would crash due to too much memory usage. At first, the amount of training data n was scaled down from 2219 to 500, however, this did not solve the problem, and thus the images were resized to

8×8, meaning that each data point has 64 variables. A visualization of this transformation for both classes can be seen in figures 2.1, 2.2. Both the train and test data set for this SVM are of length 440.

4 Results

4.1 Generated data

The values for the weights w can be found in table 1. Both the primal and dual had 14 errors, resulting in a test accuracy of 0.93. The primal was able to finish after 11 iterations, whereas the dual did so in 23 iterations using `CPLEX 20.1.0.0`.

w_1	5.32539
w_2	5.18478
w_3	5.12478
w_4	5.32085
γ	10.3291

Table 1: Weights for the primal and dual formulation using the generated data

4.2 Handwritten Digits

The values for the weights w can be found in table 2. Both the primal and dual had 1 error, resulting in a test accuracy of 0.998. The primal was able to finish after 14 iterations, whereas the dual did so in 16 iterations using `CPLEX 20.1.0.0`.

w_1	-0.000581657	w_{17}	-0.0233704	w_{33}	-0.071116	w_{49}	-0.0573293
w_2	-0.00616045	w_{18}	-0.0683274	w_{34}	-0.0255532	w_{50}	-0.0488767
w_3	-0.0373741	w_{19}	-0.0376786	w_{35}	-0.162772	w_{51}	-0.166224
w_4	-0.00256529	w_{20}	0.0105403	w_{36}	0.17299	w_{52}	0.119779
w_5	-0.0706908	w_{21}	0.0448837	w_{37}	0.112968	w_{53}	0.0082656
w_6	0.0133926	w_{22}	-0.0766765	w_{38}	-0.239913	w_{54}	-0.217565
w_7	-0.0173126	w_{23}	-0.00405228	w_{39}	-0.00988059	w_{55}	-0.0415827
w_8	0.000304213	w_{24}	-0.0360778	w_{40}	-0.0785383	w_{56}	-0.00838942
w_9	-0.0127573	w_{25}	-0.061203	w_{41}	-0.0686758	w_{57}	-0.0109158
w_{10}	-0.044201	w_{26}	-0.039754	w_{42}	-0.0341429	w_{58}	-0.0479803
w_{11}	-0.0925848	w_{27}	-0.0869442	w_{43}	-0.162544	w_{59}	-0.100162
w_{12}	-0.0578607	w_{28}	0.111616	w_{44}	0.21674	w_{60}	-0.103105
w_{13}	0.00281593	w_{29}	0.110962	w_{45}	0.115555	w_{61}	-0.090599
w_{14}	0.0130166	w_{30}	-0.179999	w_{46}	-0.194743	w_{62}	-0.0712325
w_{15}	-0.03114	w_{31}	0.00609264	w_{47}	-0.0467369	w_{63}	-0.00174136
w_{16}	-0.0139921	w_{32}	-0.0715685	w_{48}	-0.0544139	w_{64}	-2.34506e-05
γ	-1.61389						

Table 2: Weights for the primal and dual formulation using the handwritten digits data. Note that the actual weights of the dual and primal differ with at max 1e-4.