SUPPORT VECTOR CLASSIFIER IN AMPL

3rd lab assignment

Davis Bergés Lladó, Roser Cantenys Sabà

Mathematic Optimization, GCED, UPC Spring 2019

Support vector classifier in AMPL

Índex

Support vector classifier in AMPL

INTRODUCTION

The goal of this project is to implement the primal and dual quadratic formulation of the support vector machine classifier in AMPL.

A support vector machine is a supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

Given a matrix of data (A) and a response value for those observations (y), the SVM minimizes the following constrained problem:

$$\begin{aligned} \min_{\omega,\gamma,s} \quad & \frac{1}{2}\omega^T\omega + ve^Ts \\ s.to \quad & -y(A\omega + \gamma e) - s + e \leq 0 \\ & -s \leq 0 \end{aligned}$$

which is the primal problem. We can also solve the dual one:

$$\begin{aligned} \max_{\lambda} \quad & \lambda^T e - \frac{1}{2}\lambda^T yAA^T y\lambda \\ s.to \quad & \lambda^T ye = 0 \\ & 0 \leq \lambda \leq v \end{aligned}$$

Notice we optimize the variable λ and we don't get ω nor γ. We can recover them using the following calculations:

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

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

$$for\ some\ i\ such\ that\ x_i is\ a\ support\ vector$$

Support vector classifier in AMPL

AMPL CODE

PRIMAL MODEL

(SVMprimal.mod)

```
# Parameters
param n >= 1, integer; #features
param m >= 1, integer; #number of observations
param nu;          #tradeoff parameter

param y {1..m};       #response value
param A {1..m,1..n};   #feature values

# Variables
var w {1..n};
var gamma;          #intercept
var s {1..m};        #slacks


# Primal problem of the SVM
minimize primal_SVM:
        0.5*sum{j in {1..n}}(w[j]^2) +nu*sum{i in {1..m}}(s[i]);

subject to c1 {i in {1..m}}:
        -y[i]*(sum{j in {1..n}}(A[i,j]*w[j]) +gamma) -s[i] +1 <= 0;
subject to c2 {i in {1..m}}:
        -s[i] <= 0;
```

DUAL MODEL

(SVMdual.mod)

```
# Parameters
param n >= 1, integer; #features
param m >= 1, integer; #observations
param nu;          #tradeoff parameter

param y {1..m};       #response value
param A {1..m,1..n};   #feature values

# Variables
var lambda {1..m} >= 0, <= nu;

# Dual problem of the SVM
maximize dual_SVM:
        sum{i in {1..m}}lambda[i]
        -0.5*sum{i in {1..m}, j in {1..m}}lambda[i]*y[i]*lambda[j]*y[j]
    *(sum{k in {1..n}}A[i,k]*A[j,k]);

subject to c1:
        sum{i in {1..m}}(lambda[i]*y[i]) = 0;
```

Hence that our implementation uses a linear kernel. Thus, ϕ is the identity.

Support vector classifier in AMPL

OPTIMIZATION

(SVM.run)

```
reset;
model SVMprimal.mod;
data "./DATA/Extra/svm_extra_data.dat"; #gensvmdat: ./DATA/Generator/input/svm_data1.dat / extra: ./DATA/Extra/svm_extra_data.dat

option solver minos; #cplex/minos;

problem PRIMAL_SVM: w, gamma, s, primal_SVM, c1, c2;
solve PRIMAL_SVM;
display w, gamma, s;

reset;
model SVMdual.mod;
data "./DATA/Extra/svm_extra_data.dat";

option solver minos;

problem DUAL_SVM: lambda, dual_SVM, c1;
solve DUAL_SVM;
display lambda;
```

RETRIEVING ω AND γ

(SVM.run)

```
# Retrieving normal vector w from the dual solution lambda
param w {1..n};
let {j in {1..n}} w[j] := sum{i in {1..m}}lambda[i]*y[i]*A[i,j];


# Retrieving 'intercept' gamma from the dual solution lambda
# We need to find some lambda[i] that is greater than 0 but different from nu
param gamma;
param index; #auxiliary index
# Loop over the lambda vector and set the index
for {i in {1..m}} {
        if lambda[i] > 0 and lambda[i] != nu then let index := i;
}
let gamma := 1/y[index] -sum{j in {1..n}}w[j]*A[index,j];


display w, gamma;
```

PREDICTIONS

(SVM.run)

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

# We build a confusion matrix
param confusion_matrix {1..2,1..2} default 0;
for {i in {1..m}} {
		if predictions[i] == y[i] and predictions[i] = 1 then
				let confusion_matrix[1,1] := confusion_matrix[1,1] +1;
		if predictions[i] == y[i] and predictions[i] = -1 then
				let confusion_matrix[2,2] := confusion_matrix[2,2] +1;
		if predictions[i] != y[i] and predictions[i] = 1 then
				let confusion_matrix[1,2] := confusion_matrix[1,2] +1;
		if predictions[i] != y[i] and predictions[i] = -1 then
				let confusion_matrix[2,1] := confusion_matrix[2,1] +1;
}

printf "Confusion matrix of our classification problem.\n[1,1]:TP, [2,2]:TN, [1,2]:FP, [2,1]:FN \n\n";
display confusion_matrix;

param total_accuracy = (confusion_matrix[1,1] +confusion_matrix[2,2]) / m;
display total_accuracy;
```

Notice that, our implementation of the SVM only predicts on data whose response value has only two categories and has been previously mapped to -1 and 1.

It can be generalized after applying some minor changes in the prediction section of the code above.

Support vector classifier in AMPL

ANALYSIS OF RESULTS

We generated three files of sizes: 30, 100 and 1000, respectively, with the Linux executable that we were given. After applying some modifications to the file generated with an R script in order to make them comparable with AMPL we were ready to go.

SIZE 30

```
ampl: include SVM.run;
MINOS 5.51: optimal solution found.
28 iterations, objective 9.270359017
Nonlin evals: obj = 44, grad = 43.
w [*] :=
1  0.707499
2  0.378016
3  0.0241151
4  0.902912
;

gamma = -0.288688

s [*] :=
 … (omitted 30 outputs)
;

MINOS 5.51: optimal solution found.
26 iterations, objective 9.270359017
Nonlin evals: obj = 32, grad = 31.
lambda [*] :=
 … (omitted 30 outputs)
;
```

```
w [*] :=
1  0.707499
2  0.378016
3  0.0241151
4  0.902912
;

gamma = -0.288688

predictions [*] :=
 … (omitted 30 outputs)
;

Confusion matrix of our classification problem.
[1,1]:TP, [2,2]:TN, [1,2]:FP, [2,1]:FN

confusion_matrix :=
1 1    20
1 2    10
;

total_accuracy = 0.666667
```

When we use a small file size, of size 30. As we have introduced a lot of errors it doesn't provide good results. We only get a 66.67% of total accuracy. Indeed, we could improve these results using a larger file size.

In addition, we can also check that the separation hyperplane from the dual model (gamma) coincides with that of the primal model as we can see in the results given by AMPL above. Its value is, in both cases, -0.289. And the number of iterations done by each problem is very close.

SIZE 100

```
ampl: include SVM.run;
MINOS 5.51: optimal solution found.
88 iterations, objective 27.90968004
Nonlin evals: obj = 98, grad = 97.
w [*] :=
1  2.01915
2  1.24819
3  1.13926
4  1.87498
;

gamma = -2.94

s [*] :=
```

```
w [*] :=
1  2.01915
2  1.24819
3  1.13926
4  1.87498
;

gamma = -2.94

predictions [*] :=
  … (omitted 100 outputs)
;

Confusion matrix of our classification problem.
```

```
  … (omitted 100 outputs)
;

MINOS 5.51: optimal solution found.
91 iterations, objective 27.90968004
Nonlin evals: obj = 109, grad = 108.
lambda [*] :=
  … (omitted 100 outputs)
;
```

```
[1,1]:TP, [2,2]:TN, [1,2]:FP, [2,1]:FN

confusion_matrix :=
1 1   48
1 2    9
2 1    5
2 2   38
;

total_accuracy = 0.86
```

When we use a medium file size, of size 100. As we have introduced a lot of errors but we have incremented the file size, it doesn't provide optimum results but these are better than the ones obtained with a file of size 30. We get an 86% of total accuracy. We will implement the algorithm with a bigger file size to see if we can improve these results.

In addition, we can also check that the separation hyperplane from the dual model (gamma) coincides with that of the primal model as we can see in the results given by AMPL above. Its value is, in both cases, -2.94. Both arrive to the same solution with approximately the same number of iterations. However, the dual does more iterations, 2 more than the primal, and 11 more nonlinear evaluations of the objects. Thus, it should be more convenient using the primal than the dual problem in this case.

SIZE 1000

```
ampl: include SVM.run;
MINOS 5.51: optimal solution found.
1049 iterations, objective 181.2374009
Nonlin evals: obj = 1078, grad = 1077.
w [*] :=
1  2.88981
2  3.21489
3  3.09012
4  3.07831
;

gamma = -6.12691

s [*] :=
   … (omitted 1000 outputs)
;

MINOS 5.51: optimal solution found.
507 iterations, objective 181.2374009
Nonlin evals: obj = 564, grad = 563.
lambda [*] :=
  … (omitted 1000 outputs)
;
```

```
w [*] :=
1  2.88981
2  3.21489
3  3.09012
4  3.07831
;

gamma = -6.12691

predictions [*] :=
   … (omitted 1000 outputs)
;

Confusion matrix of our classification problem.
[1,1]:TP, [2,2]:TN, [1,2]:FP, [2,1]:FN

confusion_matrix :=
1 1   463
1 2    32
2 1    38
2 2   467
;

total_accuracy = 0.93
```

Support vector classifier in AMPL

Finally, when we use a large file size, of size 1000 we observe that we have achieved better results. We have an 93% of accuracy. Despite of the errors introduced, as we have really increased the file size, we get good results, s good total accuracy.

In addition, we can also check that the separation hyperplane from the dual model (gamma) coincides with that of the primal model as we can see in the results given by AMPL above. Its value is, in both cases, -6.12. Anyway, to minimize the dual problem takes half the number of iterations than the primal does So, it is more convenient to work with the dual problem in large file sizes.

APPLICATION TO ANOTHER DATASET

Now, we are going to apply our implementation of the primal and dual quadratic formulation of the support vector machine classifier in AMPL to a new data set. This one is about banknote authentication.

RESULTS

```
ampl: include SVM.run;
MINOS 5.51: optimal solution found.
2425 iterations, objective 18.95641499
Nonlin evals: obj = 2090, grad = 2089.
w [*] :=
1  -1.81349
2  -1.12435
3  -1.30482
4  -0.187653
;

gamma = 1.97985

s [*] :=
   … (omitted 1372 outputs)
;

MINOS 5.51: optimal solution found.
243 iterations, objective 18.95641499
Nonlin evals: obj = 522, grad = 521.
lambda [*] :=
   … (omitted 1372 outputs)
;
```

```
w [*] :=
1  -1.81349
2  -1.12435
3  -1.30482
4  -0.187653
;

gamma = 1.97985

predictions [*] :=
   … (omitted 1372 outputs)
;

Confusion matrix of our classification problem.
[1,1]:TP, [2,2]:TN, [1,2]:FP, [2,1]:FN

confusion_matrix :=
1 1   607
1 2    10
2 1     3
2 2   752
;

total_accuracy = 0.990525
```

As in this data set there’s no error added by us, the total accuracy achieved is bigger since in this data set the error is lower than the one added in the previous one. Its accuracy is an 99.05%, close to 100%.

In addition, in this case we also check that the separation hyperplane from the dual model (gamma) coincides with that of the primal model as we can see in the results given by AMPL above. Its value is, in both cases, 1.98.

Support vector classifier in AMPL

We can remark in this problem that the dual problem is solved with less iterations than the primal one. We observe that the primal one is solved with 2425 iterations while the dual one is solved with only 243 iterations. The dual is 10 times faster than the primal.

CONCLUSIONS

To sum up, we conclude that both, the dual and the primal problems, converge. They find the optimal solution in all the cases we have tried. However, the dual one is faster for bigger data sets. For the smallest data sets both converge with, approximately, the same number of iterations. Nevertheless, as the data set size grows the dual one converges faster than the primal one.

In addition, we check that in all examples the separation hyperplane from the dual model (gamma) coincides with that of the primal model as we have seen in the results given by AMPL above.

BIBLIOGRAPHY

- Banknote authentication Data Set, Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
- Slides of Mathematical Optimization of GCED, UPC