Lab assignment 2:

Support vector classifier in AMPL

GCED - Optimització Matemàtica

Maria Ribot Vilà & Àlex Martí Guiu

1. Introduction

The goal of this project is to apply the contents learned during the second part of the course: constrained optimization. Therefore this project consists in building a Support Vector Machine in order to classify points of a n-dimensional space into two classes. It will be done in both primal and dual formulations.

Given the implementations described above it will be necessary to check the accuracy with a different batch of the same type of data to see the level of performance of the machine built and to test the behaviour of a totally different dataset at the SVM classifier. To use the different batches we will define a training data treatment function and a test one.

Furthermore, we will have to compare the two formulations, check that their separation hyper plane is coincides and apply our SVM implementation to classify linear datasets and also a linearly non-separable dataset using a RBF or Gaussian kernel.

a. SVM Classifier

A Support Vector Machine Classifier is the application of a constrained optimization problem with the purpose to find two parallel hyperplanes separating two classes such that we both minimize the classification error and maximize the margin between the two separating hyperplanes. To do so, the implementation consists on maximizing the norm of the normal vector of the hyperplane (w), which consists on separating as much as possible the two planes we create with $w^T x + \gamma \geq +1$ and $w^T x + \gamma \leq -1$ and is equal to minimizing $\frac{1}{2} w^T w$. Therefore, our resulting two class classification of the points will be:

- Point x belongs to class 1 if $w^T x + \gamma \geq +\delta$
- Point x belongs to class 2 if $w^T x + \gamma \leq -\delta$

We have also developed our SVM so that it is able to classify linearly inseparable data, that is, by using slacks to allow misclassifications (s>0) and also by allowing the use of mapping functions that can change the input space to a feature space where the dimension can be bigger. Finally, we have developed two versions of it, one that solves the Primal version, and another one that solves the Dual version.

b. Primal SVM

The primal problem (standard one), minimizes in terms of the variables {w,gamma,s} the following function derived from the initial problem:

$$\min_{(w,\gamma,s)\in\mathbb{R}^{N+1+m}} \quad \frac{1}{2}w^\top w + \nu\sum_{i=1}^{m} s_i$$
$$\text{s. to} \quad y_i(w^\top\phi(x_i)+\gamma)+s_i \geq 1 \quad i=1,\ldots,m$$
$$s_i \geq 0 \quad i=1,\ldots,m$$

Therefore, the predictions of this primal implementation of the SVM, taking into account that matrix A is formed by the mapping functions, will be in the form of $w^T A^T + \gamma$, and we will compare them to the "solution" vector y.

This is the primal implementation in AMPL we have used as our **svm_primal.mod**:

```
# Parameters
param n >= 1, integer;
param m >= 1, integer;
param nu; # mu de les transparencies, esculls tu el seu valor
param y {i in 1..m};
param A {1..m,1..n};

# Variables
var gamma;
var w {1..n};
var s {1..m} >= 0;

# Minimitzar
minimize fobj_svm_primal: 1/2*sum{j in 1..n}(w[j]*w[j]) + nu*sum{i in 1..m}(s[i]);
subject to h{i in 1..m}: y[i]*(sum{j in 1..n}(w[j]*A[i,j])+gamma)+s[i] >= 1;
```

c. Dual SVM

The basic idea of the dual formulation of optimization is to minimize the lagrangian in terms of the lagrangian multipliers, subject to $\frac{dL}{dx} = 0$ (gradient by x of lagrangian is equal to zero). Therefore, the dual implementation of the SVM has the following formulation after having derivated the initial dual formulation:

$$\max_{\lambda} \quad \lambda^\top e - \tfrac{1}{2}\lambda^\top YAA^\top Y\lambda$$
$$\lambda^\top Ye = 0$$
$$0 \leq \lambda \leq \nu$$

To retrieve the parameters that we would have obtained with the primal solution we will use the following expressions:

- As $w = A^T Y\lambda$, we will retrieve it as $w = \sum_{i=1}^{m} \lambda_i y_i \varphi(x_i)$. Note that if we didn't have A and only had the kernel $K = A^T A$ we could still retrieve the predictions as to obtain them we need to compute $\varphi(x_j)w$ from which we can derive $\sum_{i=1}^{m} \lambda_i y_i K(x_i, x_j)$

- We will also retrieve gamma as $\gamma = \frac{1}{y_i} - w^T \varphi(x_i)$ where x_i is a point such that $s_i \neq 0$, that is, such that the point is a support vector ($\lambda_i > 0$). The same idea of only computing K can be applied to this calculus.

This is the dual implementation in AMPL we have used in our **svm_dual.mod**:

```
# Parameters
param n >= 1, integer;
param m >= 1, integer;
param nu >=0.5;

param A {1..m,1..n};
param K {1..m,1..m};
param y {i in 1..m};

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

# Minimitzar
maximize fobj_svm_dual:
    sum{i in 1..m}(lambda[i])
    -1/2*sum{i in 1..m, j in 1..m}(lambda[i]*y[i]*lambda[j]*y[j]*K[i,j]);

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

2. AMPL code & Use of python to execute

To execute the AMPL code we have designed for both implementations and retrieve the parameters & variables to afterwards use them to calculate predictions and accuracies we have used the python package “amplpy” which allows us to manage the AMPL executions from a python script where we have then designed functions to treat those executions. To run this code we have implemented we have had to have the .mod and .dat files in the same folder as the python .ipynb script.
Finally, to generate the .dat initial files we have used “gensvmdat” as a point generator.

3. Results & Analysis

a. Gensvmdat Points

The first points we have used for our classifiers have been created with the “gensvmdat” program with the seed 1234**.** From our implementation we have obtained the following results:

- SVM with the primal formulation:
  - Training accuracy: 91%
  - Test accuracy: 88.4%
- SVM with the dual formulation:
  - Training accuracy: 91%
  - Test accuracy: 88.4%

The training accuracy is measured by comparing the classification output to the true value of the sample, of the data that has been used to create the SVM. The test data has not been used in the SVM construction and gives us a measure of the performance of the newly constructed SVM applied to future similar data.
We see that both of the formulations give the same accuracies, it makes perfect sense as because of the fact that SVM is a convex problem (as we checked in theory lessons), the solution derived from both formulations should be the same. We can check that it is true by checking that the separation hyperplanes of both are equal.

w primal := [[1.73188892], [1.82189662], [1.82182107], [1.655764]]
w dual := [[1.73188877], [1.82189716], [1.82182025], [1.65576422]]
gamma primal & dual = -3.6579280069385725

The *w* coefficients of the primal and dual formulations can be considered identical.
Note: we have tried the same experiments changing the nu value (predetermined=0.5) and have gotten excellent results as well.

b. Iris Points

We can now observe the performance of our SVM given a different dataset, in our case we got the data base from the following repository: https://archive.ics.uci.edu/ml/datasets/Iris. It describes physical measures of Irises and it included the species that each observation was. From there we created three different datasets, one for each binary variables drawn from each class. We wanted to see which was the most distinctive specie and if they all behaved similarly:

Iris Type-1 → Primal & Dual Training accuracy: 100%
Iris Type-2 → Primal & Dual Training accuracy: 74%
Iris Type-3 → Primal & Dual Training accuracy: 98.67%

```
W iris1:
 [[-0.0355123 ]
 [ 0.45138075]
 [-0.87612727]
 [-0.39811229]]

 [[-0.0355123 ]
 [ 0.45138075]
 [-0.87612728]
 [-0.3981123 ]]

W iris2:
 [[ 0.00909582]
 [-1.79942868]
 [ 0.36151845]
 [-0.94013852]]

 [[ 0.00909582]
 [-1.79942868]
 [ 0.36151845]
 [-0.94013852]]

W iris3:
 [[-0.47356568]
 [-0.46601908]
 [ 1.83651599]
 [ 1.70013342]]

 [[-0.47356587]
 [-0.46601907]
 [ 1.83651578]
 [ 1.70013395]]

Gamma iris1:
 1.380028786875222
 1.380028786875222

Gamma iris2:
 4.398232853094191
 4.398232853094191

Gamma iris3:
 -7.564276080920203
 -7.564276080920203
```

As we can see with the results, both the primal and dual implementations give the same results in terms of accuracy and also in terms of the separating hyperplane (defined by w and gamma).

Our dataset is quite small so we decided to only compute the training accuracy, in order to compute the test accuracy of the dataset we should have separated a fraction of our data before building the SVM and then run it on the data left and check the respective accuracy.
We can clearly see that the second type of iris classifier is not as efficient as the other two. And we can only affirm that the type-1 iris is linearly separable from type-2 and type-3 which are not linearly separable as its training accuracy implies, they have misclassification errors. This results obtained coincide with the description of the dataset, where it's said that only one of the three types of iris is fully separable (we have guessed type-1).

c. RBF Gaussian Kernel Points

As a last task we have to generate a linearly non separable dataset (we will use the sklearn.datasets.make_swiss_roll() function of the python sklearn package to do so), which we can classify using our SVM implementation with a RBF Gaussian kernel.

When $x, y \in R^n$, we define the kernel as: $K(x,y) = e^{\frac{-\|x-y\|^2}{2\sigma^2}}$

In order to use our SVM implementation with the Kernel we have to use our implementation of the dual SVM and instead of giving the variable matrix of the data we enter K, and the targets have to be binarized. For the test set, we will have to create the Kernel by using the A matrix from the training set and the A matrix from the test set.
These are the results obtained:

Accuracy	nu = 0.1	nu = 0.5	nu = 5
Training	86%	99%	100%
Test	67%	99%	98%

Therefore, we see how the Gaussian kernel has provided us a very successful mapping transformation for the points which results in quite satisfactory classifications in the feature space of the points which in the input space were linearly inseparable.

We can observe that we have obtained a quite accurate model, as the accuracy rate is almost 100% in the posterior test run of the model. The model with the standard nu value (0.5) has proven to be a good model as its test accuracy is very high. If it were not, changing the nu values provides us with different models that could be a better fit, in this case we don't need to change it.

4. Conclusions

After the different evaluations we can conclude that the classifications given by our SVM are quite good on average. We have checked for all datasets that the hyperplane of separation is equal in the SVM dual and primal formulation, confirming the convexity of the SVM problem.

In our own dataset we have been able to identify with quite certainty the species of iris each sample belongs too, finding type1 easiest one to classify properly.

We have established that even when we start off linearly non-separable data we can use a Kernel transformation in order to properly classify the data. In our case the Gaussian Kernel is evidently quite efficient.

5. References

- The formulations of both primal and training problems have been derived from the theory files of OM
- All the results are replicable with the code attached with this report in the .zip file, by executing the .ipynb file having all .mod and .dat files in the same folder