Support Vector Classifier

AMPL implementation

Jordi Puig Rabat
Marc Vernet Sancho
OPTIMITZACIÓ MATEMÀTICA
UPC 2020

Introduction

The main goal of this project is to design and implement a Support Vector Machine (SVM) in AMPL. The SVM will be implemented in primal and dual form. Some artificially generated datasets will be used to check the accuracy and performance of the SVM. In addition, a real dataset will be used.

SVM are supervised learning models for classification problems. Starting with a training dataset, where each element is categorized in one of two categories, SVM builds a model that assigns new samples to one or the other category.
Given *n* points in the d-dimensional space, each one belonging to one of the two groups, the purpose of the SVM is to find the hyperplane separating the two classes such that we minimize the classification error and maximize the separation of the two semi planes defined as:

$$w^T x + \gamma \geq +1$$
$$w^T x + \gamma \leq -1$$

Where the separation hyperplane is defined by the normal vector *w* and the distance respect to the origin ɣ.

SVM Primal

The primal formulation has as objective function the sum of the scalar product of the normal vector of the plane and the weighted sumatory of the errors. Minimize the scalar product is equivalent to maximize the separation between the two semi planes explained before. It is subject to the condition that every point of one class belongs to one of the semi planes.
Here we have the formulation of the problem in the standard notation:

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

$$s.\ to\ \ y_i(w^T x_i + \gamma) + s_i \geq 1 \qquad i = 1,\ldots,m$$

$$s_i \geq 1 \qquad i = 1,\ldots,m$$

Where *w* is the normal vector of the plane, s_i are the slacks to account classification error for every point and ɣ is the parameter that determines the location of the plane respect the origin. With the slack variables s we are considering linearly non-separable data.

We write this formulation in AMPL with the next objective function and this constraints:

```
minimize objective_func:
 (1/2) * sum{n in 1..numVar}w[n]^2 + v*sum{m in
1..numElements}s[m];

subject to constr {m in 1..numElements}:
    y[m]*((sum{n in 1..numVar}w[n]*x[m,n]) + gamma) + s[m] >= 1;
```

SVM Dual

The SVM is a convex problem so if we solve the dual formulation, the optimal solution is found. As we know, the dual formulations consists in maximizing the Lagrangian function. The formulation in standard notation is:

$$\max_{(w,\gamma,s,\lambda,\mu)} \frac{1}{2} w^T \cdot w + \nu e^t s + \lambda^T (-Y(Aw + \gamma e) - s + e) - \mu^T s$$
$$s.\ to \quad w - (\lambda^T Y A^T) = 0$$
$$\lambda^T Y e = 0$$
$$\nu e - \lambda - \mu = 0$$
$$\lambda \geq 0,\ \mu \geq 0$$

Simplifying the formulation of the problem is useful for the implementation in AMPL. The simplified formulation using vectorial notation is:

$$\max_{\lambda} \lambda^T e - \frac{1}{2} \lambda^T Y A A^T Y \lambda$$
$$\lambda^T Y e = 0$$
$$\nu e - \lambda - \mu = 0$$
$$0 \leq \lambda \leq \nu$$

In AMPL it is written this way:

```
maximize objective_func: sum{m in 1..numElements}
  lambda[m] - (1/2)*sum{mi in 1..numElements, mj in 1..numElements}
  lambda[mi]*y[mi]*lambda[mj]*y[mj]*K[mi,mj];

subject to constr:
    sum{m in 1..numElements} lambda[m]*y[m] = 0;
```

Another simplification has been made in the implementation, it is possible to assign the value of μ and then consider the last constraint.

$$\mu = \nu e - \lambda$$

The Kernel matrix:

There is an element `K[mi, mj]` in the implementation of the dual formulation which refers to the *mi,mj*-th element of the matrix K. Given two points x_{mi}, x_{mj} the *mi,mj*-th element of the matrix is the computation of the kernel. Two different kernels can be used:

The identity kernel:

$$K(x, y) = x^T y$$

The gaussian kernel:

$$K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

In the AMPL implementation, the gaussian kernel is used. The element $-\frac{1}{2\cdot\sigma^2}$ can be approximated as $-\frac{1}{\gamma}$, where γ is the number of features.

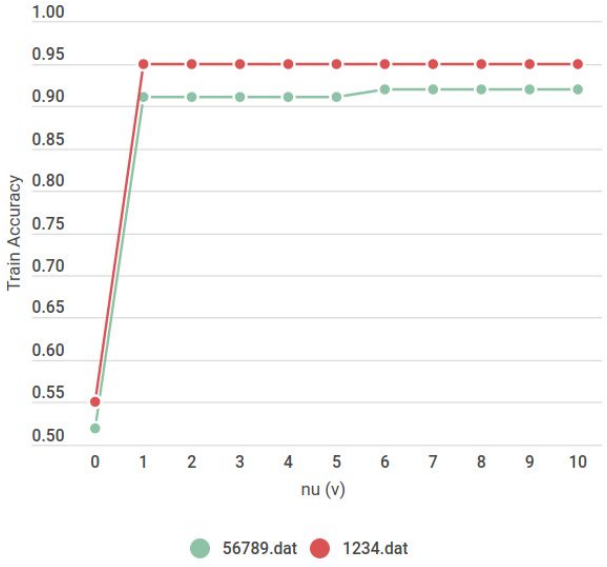
Performance

The datasets used to train the models and check the performance are generated with the executable script *gensvmdat* facilitated as part of the documentation of the project. This generated datasets are composed of 100 points (rows of the dataset), each one with 4 variables and a value that denotes to which class the point belongs (marked as 1 or -1).

To validate the results of the SVM and know the accuracy, is necessary to use the values of ω, v, and ɣ and s obtained, using the formula:

$$\omega^T x + \gamma = 0$$

If the result of the formula is bigger than 0, the element is categorized as 1, otherwise it's categorized in the -1 class.
The value of ω and ɣ are generated with the training of the SVM, but the variable v that has to be specified by the user. In order to evaluate the performance we need to determine which is the best v value. The best value uses to be between 1 and 10.
For these test, the datasets used are the ones generated with the seeds *56789* and *1234*. The next graph shows the value of the train accuracy over the parameter *nu.*



It seems that any value between 1 and 10 will give good enough results. We will use v = 10. Using v=10, for the first dataset we have a 92% train accuracy and for the second one, 95%.

In order to know the test accuracy, it's necessary to predict a test dataset with a model created from a train dataset. In this case, the model has been trained with the dataset with seed *1234.* Using v=10 and applying the values ω and ɣ found in the

training of the model to the dataset of seed *56789*, the result is a 92% of test accuracy. Really good results, considering that the model created with *56789* as training dataset gave very similar results.

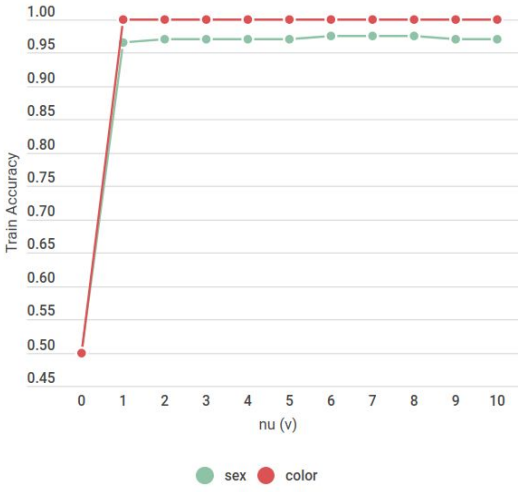
An important characteristic of SVM is that both dual and primal models should give the same output. This can be checked with the data output of the same execution of the *1234* seed dataset with v=10:

Primal	**Dual**
122 iterations, objective 276.403539 Nonlin evals: obj = 124, grad = 123. numCorrect/numElements = 0.95	147 iterations, objective 276.403539 Nonlin evals: obj = 261, grad = 260. numCorrect/numElements = 0.95
w [*] := 1 4.48905 2 3.78091 3 4.53887 4 4.0448;	w [*] := 1 4.48905 2 3.78091 3 4.53887 4 4.0448;
gamma = -8.39446	gamma = -8.39446

This proves our implementation is correct.

Crabs dataset

To further test the accuracy, we will use a real linearly non-separable dataset. We have a dataset of 200 rows and 8 columns, describing 5 morphological measurements on 200 crabs of two color forms and both sexes of the species Purple Rock Crab (*Leptograpsus variegatus*). In order to use the SVM for the classification, is necessary to classify in two categories. We will realize two experiments, in the first, the crabs will be going to be classified by color, ignoring the sex. In the second they will be classified by sex, ignoring the color. Classifying by color is very easy, it's near linearly separable data. Classifying by sex is more difficult, as there's more overlapping and there isn't a clear gap.
The next graph shows the value of the train accuracy over the value of the parameter *nu*.



When using a different dataset for train and test and v=10, the test accuracy found is 99% for color classification and 96% for sex classification. Very good results in comparison with the train error.

Conclusion

After all the test, it can be proven the efficiency and good performance of SVM implemented in AMPL, at least using this datasets. The test accuracy maintains similar levels with the train accuracy. Besides that, it has been proven that dual and primal models give the same result.

In all the models used, the execution speed has been very fast, without difference between dual and primal models. This is probably due to the fact that the datasets used where rather small, as it's known that with large datasets SVM methods can become slow.

Annex

SVM.dat : AMPL .dat file used for primal and dual models training
dataset_SVM.dat : AMPL .dat file used for training the model of the crabs dataset
testSVM.dat : AMPL .dat file used for primal and dual models test accuracy

dualSVM.mod : AMPL dual model
dualSVM.run : script for executing the dual model and calculation of accuracy

primalSVM.mod : AMPL primal model
primalSVM.run : script for executing the primal model and calculation of accuracy

primalTestSVM.mod : AMPL primal model (with variables for the test dataset)
primalTestSVM.run : script for executing the primal model and calculation of test accuracy using a test dataset

SVM_dataset.Rmd : R markdown script for converting the crab dataset to a format accepted by AMPL.