Support Vector Machines
Optimization
DMKM

Carlos López Roa
me@mr3m.me

December 17, 2015

Abstract

We used a Support Vector Machine (SVM) to train a binary classifier.

Introduction

Given a m vectors in $\mathbb{R}^n$ represented in the matrix $A \in \mathbb{R}^{m\times n}$ labeled by the binary vector $\hat{y}$ with $\hat{y}_i \in \{-1, 1\}$, we can construct the *linear kernel support vector machine* with the quadratic optimization problem:

Problem 1

$$\begin{aligned} \min_{\omega,\gamma,y} \quad & \nu e^T y + \frac{1}{2}\omega^T\omega \\ s.t. \quad & D \times (A\omega - e\gamma) + y \geq e \\ & y \geq 0. \end{aligned} \tag{1}$$

With $\omega \in \mathbb{R}^n$, $\gamma \in \mathbb{R}$, $y \in \mathbb{R}^m$, $e = \mathbf{1} \in \mathbb{R}^m, \nu > 0$ *and* $D = \operatorname{diag}(\hat{y})$.

Recall $||x||_2^2 = \langle x, x\rangle = x^T x$. The solution of problem 1 defines a hyperplane

$$x^T\omega = \gamma, \tag{2}$$

which separate the points represented in matrix A, into two semispaces.

Thus we are to evaluate the error of the classifier using the formula

$$\epsilon = \frac{1}{2m}\sum |\operatorname{sign}(A\omega - \gamma) - \hat{y}| \tag{3}$$

Implementation

We chose to implement the `CVX` solver in `Matlab 2011a 7.12.0.635`.

First we generated the data using the provided binary of sizes $10^1, 10^2, 10^3, 10^4$, with seed `26071991`.

Then we parsed the output into `txt` files using the supplied code in python, removing the special cases marked with `*` as follows

```python
i=1
f1 = open('x10.txt','w')
f2 = open('y10.txt','w')
with open('10.dat','r') as rf:
    reader = csv.reader(rf,delimiter=' ')
    for row in reader:
        if "*" not in row[6]:
            f1.write(row[0]+' '+row[1]+' '
                +row[2]+' '+row[3]+'\n')
            f2.write(row[6]+'\n')
        i += 1
f1.close()
f2.close()
```

The following is to call the solver in `Matlab`

```matlab
err = []; temp =[]; i=3;
At=importdata('x100000.txt');
ytt=importdata('y100000.txt');
sizes=[10,100,1000,10000,100000];
A=At(1:sizes(i),1:4);
yt=ytt(1:sizes(i));
[m n]=size(A); nu=2^(7-10); e=ones(m,1);
cvx_begin quiet
    variables w(n) gam y(m)
    minimize (nu*e'*y+(0.5)*w'*w)
      subject to
      diag(yt)*(A*w-e*gam) + y >= e
      y >= 0
cvx_end
```

Where we used the best ν we could find discussed in the results.

Results

To choose the best ν in order to minimize the error trying not to overfit, we explored the parameter iterating on the range $\{2^i\}_{i=-9}^{10}$ exhibiting the result shown in figure 1. We chose the minimum ν in the *sweet* zone, that is $\nu = 2^{-3}$

Using this parameter we chose to train a 10^3 sample size and test in a 10^4 sample size, that is training with just 10% of the sample.

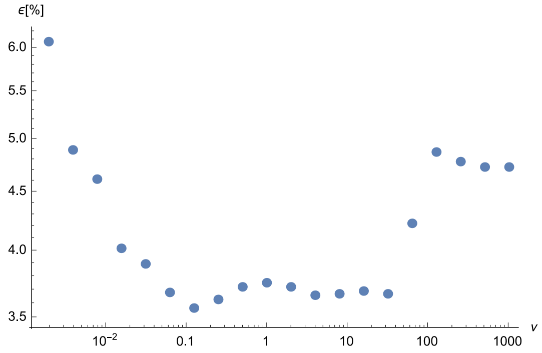

Figure 1: Behaviour of the error as a function of the parameter ν we observe a *sweet* zone in which the error stays low, before going up again.

The results as follows:

m	$\epsilon[\%]$
10^3	4.8
10^4	5.0

That is, the sample set exhibited 95.2% accuracy and the test set 95% using only 10% of the set to train.

Conclusions

1. We implemented a linear kernel suppor vector machine to classify a binary labeled set in $\mathbb{R}^4$
2. We explored the normalization parameter of the model to minimize the error still avoiding overfitting.
3. We trained the model using only 10% of the test set and got accuracy of 95%