Support Vector Machine

Gabriel C. Leite[1]

***Abstract*— This works applies a review for a set of machine learning implementation. Considering that the demand of data analyses is increasing more and more, the mastery of techniques, like supervised learning methods, is an indispensable skill for those who want to work in this scenery. As a matter of fact, the mathematical understand, the algorithm implementation and the comparison between a set of methods will be explored in this works.**

***Keywords*— Machine learning implementation applied to computational intelligence skills.**

I. Support Vector Machine

A. INTRODUCTION

The Support Vector Machine is a supervised learning method that provides the classification, regression and outliers detection for multiple datasets.

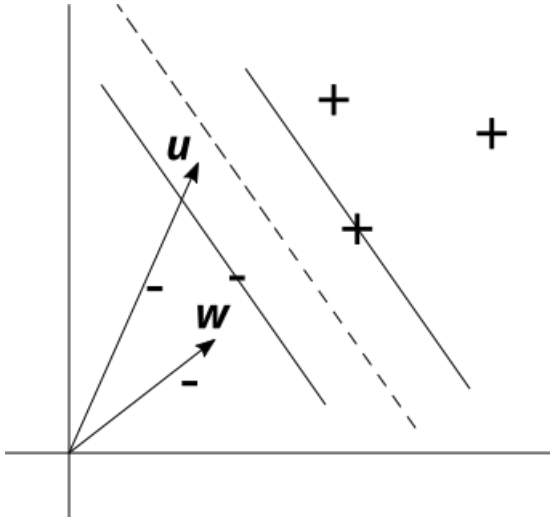
This methods applied the concept of choose the best hyperplane for divide correctly the dataset. It's implementation seeks to find the closets dataset points from the hyperplane that maximize the margin between them.

It is possible to take as advantages of support vector machines the effective in high dimensional spaces, even in cases where the number of dimensions is greater than the number of samples, the memory efficient provide by the support vectors and the versatile of the decision function modeling, which can implements different Kernel functions to obtain different models inside this method.

Otherwise, the over-fitting in cases of a number of features much greater than the number of samples and the fact that the decision function gives per-class scores for each sample, increasing the validation price, are some of the disadvantages of support vector machines.

B. MATHEMATICAL MODEL

The idea of support vector machine is to find the best hyperplane that provides the best function decision for the classification of a dataset trained by a class of training features and samples. Let's take a hypothetical dataset of $+$ and $-$:



The vector $\overrightarrow{w}$ is normal to the hyperplane chosen, so it is possible to find out the component of $\overrightarrow{u}$ in the normal direction and set that if it is greater than some value c this point will be classified as $+$:

$$\overrightarrow{w} \cdot \overrightarrow{u} \geq c$$

Without loss of generality ($c = -b$):

$$\overrightarrow{w} \cdot \overrightarrow{u} + b \geq 0 \tag{1}$$

Even so, will be applied some constraints to decision rule function:

$$\begin{cases} \overrightarrow{w} \cdot \overrightarrow{x_+} + b \geq 1 \\ \overrightarrow{w} \cdot \overrightarrow{x_-} + b \leq -1 \end{cases} \tag{2}$$

For mathematical convenience:

$$y_i = \begin{cases} +1 \rightarrow \text{ sample } + \\ -1 \rightarrow \text{ sample } - \end{cases} \tag{3}$$

Accordingly, multiplying the equation 2 by y_i and $-y_i$ gives:

$$\begin{cases} y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1 \\ y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1 \end{cases} \tag{4}$$

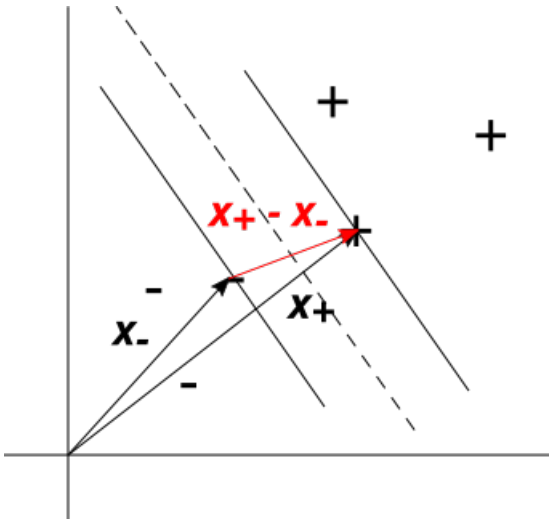
Likewise, the equations are equals and finally the equation that defines the model constraint is:

$$y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) - 1 \geq 0 \tag{5}$$

It is possible to add on more constraint to the model:

$$y_i(\overrightarrow{w}\cdot\overrightarrow{x_i}+b)-1=0 \tag{6}$$

So, the best hyperplane will be the one that maximize the distance between the closest sample + and sample −.



To maximize this distance:

$$d=(\overrightarrow{x_+}-\overrightarrow{x_-})\cdot\frac{\overrightarrow{w}}{||w||} \tag{7}$$

Replacing 6 at 7:

$$d=(1-b-(1+b))\cdot\frac{\overrightarrow{w}}{||w||}=\frac{2}{||w||} \tag{8}$$

The idea is to find the greater value for d, in other words:

$$max(\frac{2}{||w||})\therefore max(\frac{1}{||w||})\therefore min(||w||)\therefore min(\frac{1}{2}||w||^2) \tag{9}$$

Then, the real problem is to find the minimum value of $\frac{1}{2}||w||^2$

Indeed, the optimization of this equation needs to honor with the constraints. In this case, the Lagrange Multipliers gives that:

$$L=\frac{1}{2}||w||^2-\sum\alpha_i[y_i(\overrightarrow{w}\cdot\overrightarrow{x_i}+b)-1] \tag{10}$$

Optimizing L over $\overrightarrow{w}$ and b:

$$\begin{aligned}\frac{\partial L}{\partial\overrightarrow{w}}&=\overrightarrow{w}-\sum\alpha_i y_i\overrightarrow{x_i}=0\therefore\overrightarrow{w}=\sum\alpha_i y_i\overrightarrow{x_i}\\ \frac{\partial L}{\partial\overrightarrow{d}}&=\sum\alpha_i y_i=0\end{aligned} \tag{11}$$

Replacing 11 at 10:

$$L=\frac{1}{2}(\sum\alpha_i y_i\overrightarrow{x_i})\cdot(\sum\alpha_j y_j\overrightarrow{x_j})-(\sum\alpha_i y_i\overrightarrow{x_i})\cdot(\sum\alpha_j y_j\overrightarrow{x_j}) \tag{12}$$

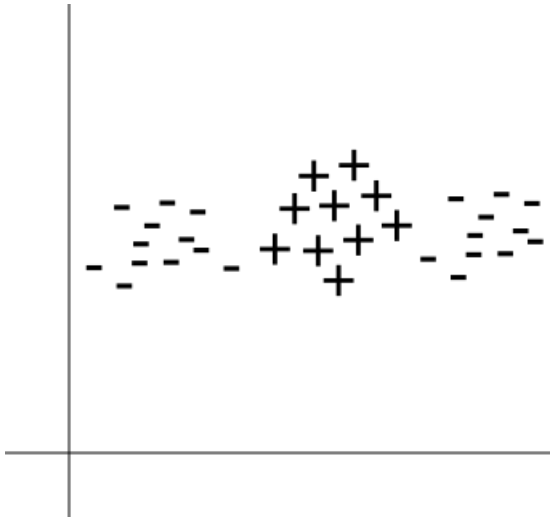
$$-\sum\alpha_i y_i b+\sum\alpha_i \tag{13}$$

$$L=\sum\alpha_i-\frac{1}{2}\sum_i\sum_j\alpha_i\alpha_j y_i y_j\overrightarrow{x_i}\cdot\overrightarrow{x_j} \tag{14}$$

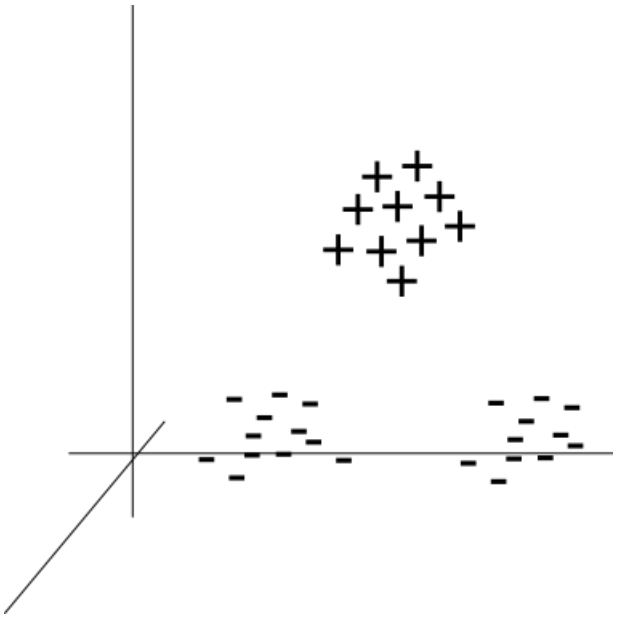
The decision function will be:

$$\sum\alpha_i y_i\overrightarrow{x_i}\cdot\overrightarrow{x_j}+b\geq 0\rightarrow\ \text{sample}+ \tag{15}$$

In the other side of the coin, this decision function only works in cases where the dataset is easily divide, but when the dataset is like:



It is impossible to find a proper hyperplane. To solve this problem, a linear transformation on the dataset is require:



This type of transformation can be solved applying the ϕ function to 15:

$$\sum\alpha_i y_i\overrightarrow{\phi(x_i)}\cdot\overrightarrow{\phi(x_j)}+b\geq 0\rightarrow\ \text{amostra}+ \tag{16}$$

Furthermore, let's call $K(x_i,x_j)=\overrightarrow{\phi(x_i)}\cdot\overrightarrow{\phi(x_j)}$. The addition of this K function is call the Kernel Trick:

One of the best Kernel function is the Radial Base Function (RBF):

$$K = e^{-\gamma(x_i - x_j)^2} \tag{17}$$

This type of Kernel transform our dataset in a infinity higher dimension, and it is possible to prove that with the Taylor Series.

Finally, the decision function will be given by:

$$\sum_{i \in SV} \alpha_i y_i K(x_i, x_j) + b \tag{18}$$

Where SV are all the support vectors.

Besides all of that, the problem isn't solve, since the α and b are unknown. This is a optimization problem to find the support vector which are also unknown.

Some papers provides a solution to find out those unknown variables. The CS229 Simplified SMO Algorithm [1] describes a simplified version of the Sequential Minimal Optimization (SMO) algorithm for training support vector machines. Others techniques can be found at LIBSVM [2].

$$max_\alpha L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{19}$$

$$0 \leq \alpha_i \leq C \tag{20}$$

$$\sum \alpha_i y_i = 0 \tag{21}$$

The Karush-Kuhn-Tucker conditions can be use to solve the optimization problem, where:

$$\alpha_i = 0 \rightarrow y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1 \tag{22}$$

$$\alpha_i = C \rightarrow y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \leq 1 \tag{23}$$

$$0 < \alpha_i < C \rightarrow y_i(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) = 1 \tag{24}$$

Every $\alpha that satisfy those conditions will be a optimal solution.$

Indeed, b must be an optimal value too. With the optimal $\alpha is possible to find the best$b.

So, choosing the Lagrange multipliers $\alpha_i and \alpha_j to optimize, let's compute the first error$E$_i$:

$$E_i = f(x_i) - y_i \tag{25}$$

where $f(x_i)$ is the decision function for x_i

Setting a tolerance error tol, if $y_i E_i < -tol$ and $\alpha_i < C$, or, $y_i E_i > tol$ and $\alpha_i > C$:

Select $j \neq i$ randomly, calculates the E_j:

$$E_j = f(x_j) - y_j \tag{26}$$

Calculate η:

$$\eta = 2(x_i \cdot x_j) - (x_i \cdot x_i) - (x_j \cdot x_j) \tag{27}$$

Now, let's find the bounds L and H such as $L \leq \alpha_j \leq H$

$$y_i \neq y_j \rightarrow L = max(0, \alpha_j - \alpha_i) \ , H = min(C, C + \alpha_j - \alpha_i) \tag{28}$$

$$y_i = y_j \rightarrow L = max(0, \alpha_j + \alpha_i - C) \ , H = min(C, \alpha_j + \alpha_i) \tag{29}$$

Thus, α_j is computed by a series of two equations, where:

$$\alpha_j = \alpha_j - \frac{y_j(E_i - E_j)}{\eta} \tag{30}$$

$$\alpha_j = \begin{cases} H \rightarrow \text{ if } \alpha_j > H \\ \alpha_j \rightarrow \text{ if } L \leq \alpha_j \leq H \\ L \rightarrow \text{ if } \alpha_j < L \end{cases} \tag{31}$$

With $\alpha_j, let's determine \alpha_i$:

$$\alpha_i = \alpha_i + y_i y_j (\alpha_j^{old} - \alpha_j) \tag{32}$$

Finally, it is possible to compute b:

$$b = \begin{cases} b_1 \rightarrow \text{ if } 0 < \alpha_i < C \\ b_2 \rightarrow \text{ if } 0 < \alpha_j < C \\ \frac{b_1 + b_2}{2} \rightarrow \text{otherwise} \end{cases} \tag{33}$$

where:

$$b_1 = b - E_i - y_i(\alpha_i - \alpha_i^{old})(x_i \cdot x_i) - y_j(\alpha_j - \alpha_j^{old})(x_i \cdot x_j) \tag{34}$$

$$b_2 = b - E_j - y_i(\alpha_i - \alpha_i^{old})(x_i \cdot x_j) - y_j(\alpha_j - \alpha_j^{old})(x_j \cdot x_j) \tag{35}$$

If $L = H$ or $\eta \geq 0$ or $|\alpha_j - \alpha_j^{old}$ repeats for a bunch of interactions, the method converge to find out the optimal α and b.

One important consideration is that the Support Vector Machine method provides only binary classification. So if the dataset have more than two samples, it can be applied the one-vs-one or one-vs-all techniques.

The one-vs-one technique takes all the combination of samples and classifier the dataset. With all classification for each combination is possible to take the mode of all of them and obtain the results.

In this studied, the regression mathematical model will not be explore, but the regression decision function is the following one:

$$\sum(\alpha_i - \alpha_i*)K(x_i, x_j) + b \tag{36}$$

C. ALGORITHM IMPLEMENTATION

One of the best way to implement the support vector machine is using the scikit-learn [3], which is a Machine Learning repository in Python.

On the other hand, the author of this paper provides a Python implementation from zero, with all mathematical implementation and example result. It can be found in:

`https://github.com/gabrielclwq/Support-Vector-Machines/blob/main/svm_smo`

D. EXPERIMENTAL RESULTS

From the scikit-learn [3], the Iris Dataset will be use to compare the author algorithm with the scikit-learn SVM model.

The division of training and test data were made randomly.

The accuracy of author's algorithm is: 0.9778

The accuracy of sklearn's algorithm is: 0.9778

Fig. 1: Author's algorithm

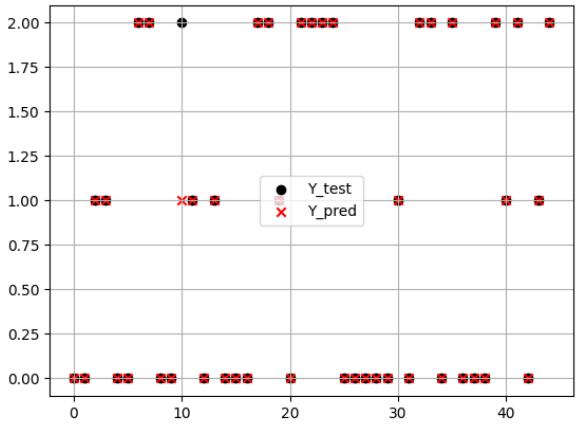
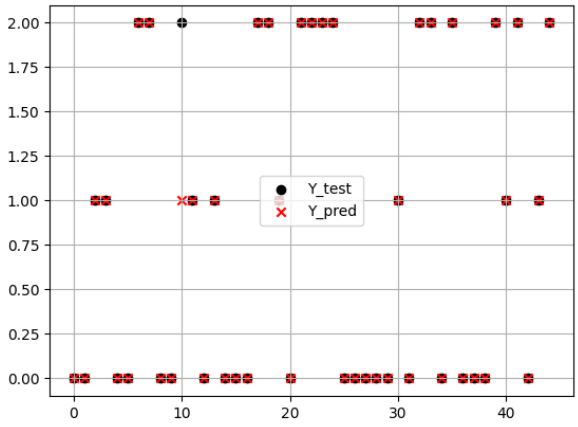


Fig. 2: Sklearn's algorithm



Also, it is possible to compare the decision surface and the support vector from both algorithm:

Fig. 3: Author's algorithm decision surface

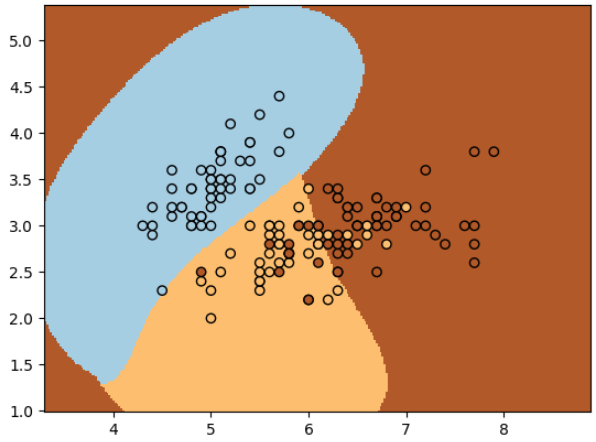
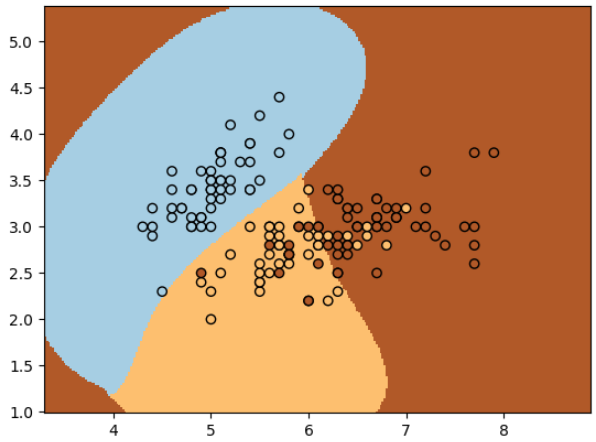


Fig. 4: Sklearn's algorithm decision surface



E. CONCLUSION

As conclusion, it is possible to see the efficiency of the Support Machine Vector method implementation from all the math explored in this paper. Also, this method applies not only a mathematical function decision, but it's principal point is the optimization problem to find out the optimal support vector from the dataset. Indeed, the Kernel Trick is extremely crucial to compute the dimension transformation in dataset with having to apply a transformation function, which would drop the algorithm efficiency of this method. An important point to cite is that even both of the results from the author's algorithm and the scikit-learn gives the same result, the author's method takes a bunch of time more than the scikit-learn one. Even all those things said, the SVM is a greater classification method.

ACKNOWLEDGMENT

This study was propose by the course of Applied Computational Intelligence in the Federal University of Ceara taught by Professor Victor Hugo Costa de Albuquerque.

REFERENCES

[1] A. Ng, "Cs229 lecture notes - supervised learning," 2012.

[2] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.