CS584 Assignment 3: Report

Cong Liu
A20343692
Department of Computer Science
Illinois Institute of Technology
April 14, 2016

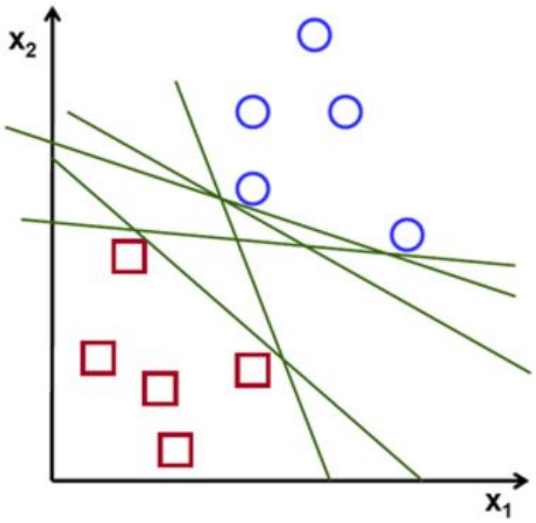
1. Abstract

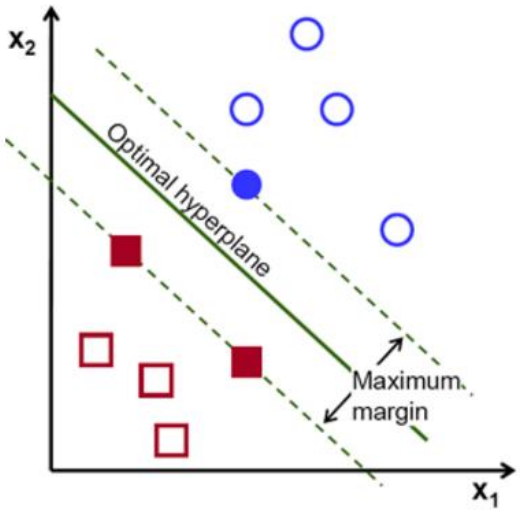
This is the report of CS584 assignment 4. In this assignment, I implement support vector machine using different kernel functions. The program and results analysis are shown in HW4.html file.

2. Problem statement

Support vector machine is a generation of learning algorithms. Before 1980, almost all learning methods learned linear decision surfaces. In 1980's, decision trees and NNs allowed efficient learning of non-linear decision surfaces. After 1990's Efficient learning algorithms for non-linear functions based on computational learning theory developed, nice theatrical properties come out. [1]

The basic method of support vector machines is to find the largest margin between two classes.





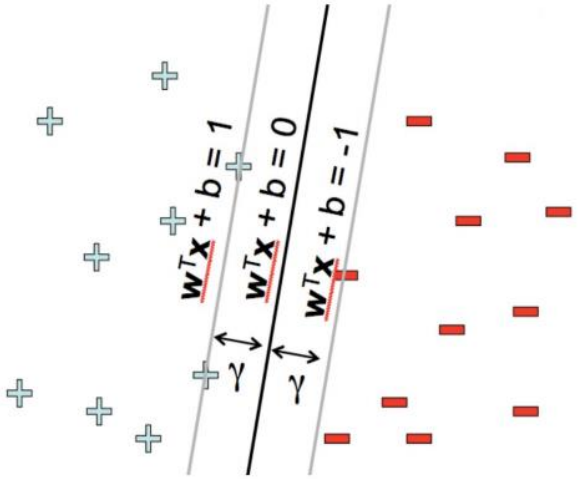
In a 2-class classification, $x \in R^n$, $y \in \{\pm 1\}$.

In training sets, the function to define a formally hyperplane is $f(x) = \beta_0 + \beta^T x$, where beta is known as the weight of features and beta0 is bias. [2]

The optimal hyperplane can be represented in an infinite number of different ways by scaling hyperplane, while margin is chosen as 1, that is: $|\beta_0 + \beta^T x| = 1$.

The training examples that are closest to the hyperplane are called support vectors.

For hard-margin SVMs, all points must be out of the margin, then the maximize margin is: $\max_w \gamma = \frac{a}{||w||}$, while a = 1.



Then we find the primal function to minimize an objective in:

Lp = $\frac{1}{m}\sum_{i=1}^{m}\ell(w\cdot x_i+b,y_i)+||w||^2$, such that $y_i(w\cdot x_i+b)\geq 1$. To minimalize Lp, we should set Lp's derivatives to zero. And solve it for w and beta.

Using Lagrange Multipliers, we could change primal problem into dual problem. In dual, W is a linear combination of training examples, $w=\sum_{l=1}^{M}\alpha_l y_l x_l$, in this case, the value of alfas would be 0 except for support vectors.

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{m}\alpha_i\left[y^{(i)}(w^Tx^{(i)}+b)-1\right].$$

$\max_{\alpha}\min_{w,b}L(w,b,\alpha)$, while $\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_N)^T$.

Using derivatives on Ld, we got:

$$\nabla_w\mathcal{L}(w,b,\alpha)=w-\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}=0 \quad , \quad \frac{\partial}{\partial b}\mathcal{L}(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i y^{(i)}=0.$$

And finally, this dual problem defines like below:

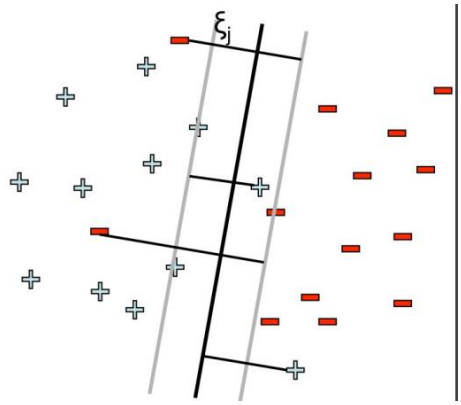
$$\begin{aligned}\max_{\alpha}\quad & W(\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}y^{(i)}y^{(j)}\alpha_i\alpha_j\langle x^{(i)},x^{(j)}\rangle.\\ \text{s.t.}\quad & \alpha_i\geq 0,\quad i=1,\ldots,m\\ & \sum_{i=1}^{m}\alpha_i y^{(i)}=0,\end{aligned}$$

After compute alfa, it is straightforward to find optimal value for w and b using $w=\sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}$,

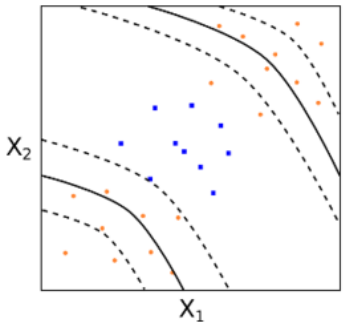
$$b^*=-\frac{\max_{i:y^{(i)}=-1}w^{*T}x^{(i)}+\min_{i:y^{(i)}=1}w^{*T}x^{(i)}}{2}.$$

For soft margin, we use slack variables ξ which represents how many 'wrong' examples could be allowed during predictions. C >0, Cis chosen by user.

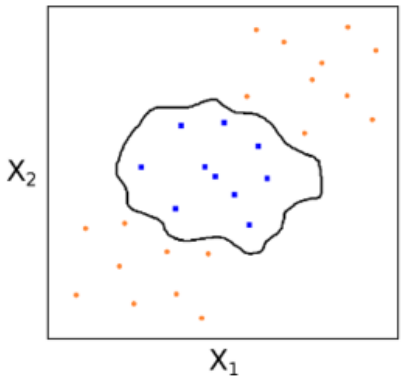
$$\min_{w}\|w\|+C\sum_{j}\xi_j$$
$$(w^Tx_j+b)\,y_j\geq 1-\xi_j$$



However, sometimes the dataset cannot be separated by linear functions. In this case, we could use some kernel functions $K(x,z) = \phi(x)^T\phi(z).$ to solve this problem. Polynomial kernel functions $K(x,z) = (x^Tz + c)^d$ can map data into higher dimension to get classification. Showing as below:



And also Gaussian kernel function $K(x,z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right).$ works well too.



As before, we can form the Lagrangian:

$$\mathcal{L}(w, b, \xi, \alpha, r) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i \left[y^{(i)} (x^T w + b) - 1 + \xi_i \right] - \sum_{i=1}^{m} r_i \xi_i.$$

and solve the following dual form of the problem:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

3. Proposed solution

In this assignment, it is a little difficult when implementing SVM using cvxopt to get alfa. This library is very strict on types of inputs. I mixed with cvxopt matrix with numpy matrix at the beginning, and cvxopt error reports are very ambiguous, usually gives more than one probabilities on errors. So it is very hard to debug.

Another difficulty is to plot the support vectors and hyperplane. I tried plt.plot, plt.scatter etc. several different methods to find the best way to solve this problem.

Also, at first I used make_moons, make_circles, make_classification from sklearn.datasets to generate datasets. While for hard margin, make_classification didn't work well because hard margin does not fit for error examples. So I used another method called np.random.multivariate_normal to make a better dataset.

According to the order of this assignment, I implement linear SVM with hard and soft margins first. However, when I finished the 4th and 5th questions, I think I could combine linear functions and kernel functions together using only one method with different parameters. So I overwrite most of my previous code to get a more efficient function.

4. Implementation details

I used make_moons, make_circles, make_classification from sklearn.datasets and np.random.multivariate_normal from numpy to generate dataset. All datasets have default 200 examples; each example has 2 features.

And all examples are belonging to two classes: {y = 1} and {y = -1}. The number of examples in each class are equal and are 100.

In these 4 functions, make_classification and np.random.multivariate_normal can implement linear-separable datasets, while make_moons, make_circles can implement non-linear separable datasets.

When implementing linear SVM and kernel-based SVM algorithms, I use only one function to compute alfa, w, and b, using different parameters to simplified my code with higher efficient. And linear SVM with soft margins and kernel-based SVM algorithms also requires user to choose the value of C. So I I tried several Cs between [0.01 to 1000], and plot the support vectors and hyperplane to find out the best choice of C.

5. Results and discussion

Results and discuss are shown in the coding attachments.

Demonstrations of correctness are also showed in attachments.

6. References

[1]. An Idiot's guide to Support vector machines (SVMs), R. Berwick, Village Idiot, 2003

[2]. Elements of Statistical Learning by T. Hastie, R. Tibshirani and J. H. Friedman.

[3]. Introduction to Support Vector Machines, OpenCV 2.4.12.0 documentation: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[4]. Pattern Recognition and machine learning, Christopher M.Bishop

[5]. UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/