SUPPORT VECTOR MACHINE

PART 1 IMPLEMENTATION

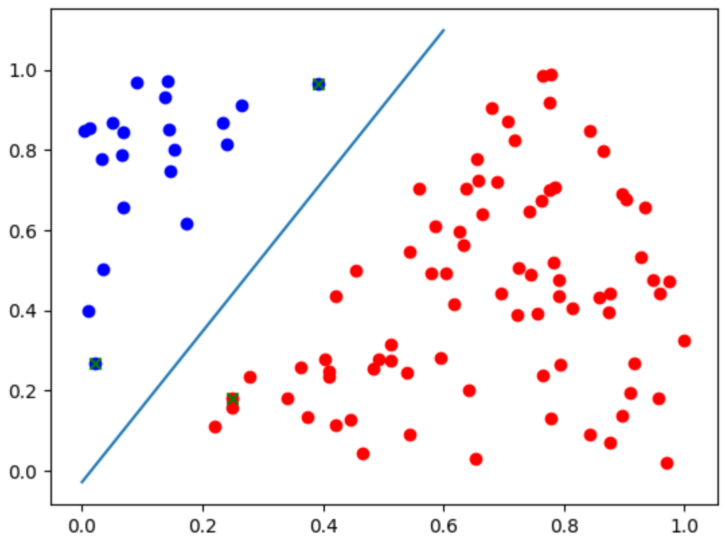
The implementation of this problem begins with finding a quadratic solver that would be able to solve the maximization quadratic programming for this problem. I began looking for a good choice for python, and found the cvxopt package. However, many of the quadratic programming packages (including this one) will only allow standard form of qp:

$$\begin{aligned}&\min x^T P x + p^T x\\ &s.t.:\\ &Gx \leq h\\ &Ax = b\end{aligned}$$

Since our problem is a maximization, we must negate our objective function. Thus, our problem becomes:

$$\begin{aligned}&\min x^T Q x - \begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix} x\\ &s.t.:\\ &-Ix \leq [0,0 \dots 0]\\ &\begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix} x = 0\end{aligned}$$

We run the optimization, to obtain our alphas (in the formula above, x). The plot of the line, as well as each point is the following figure. The support vectors (the vectors with alpha values that are sufficiently not close to zero) are marked with green. (there are only 3)



In the output, we see that:

W = [7.250056295527888, -3.861889241774133]

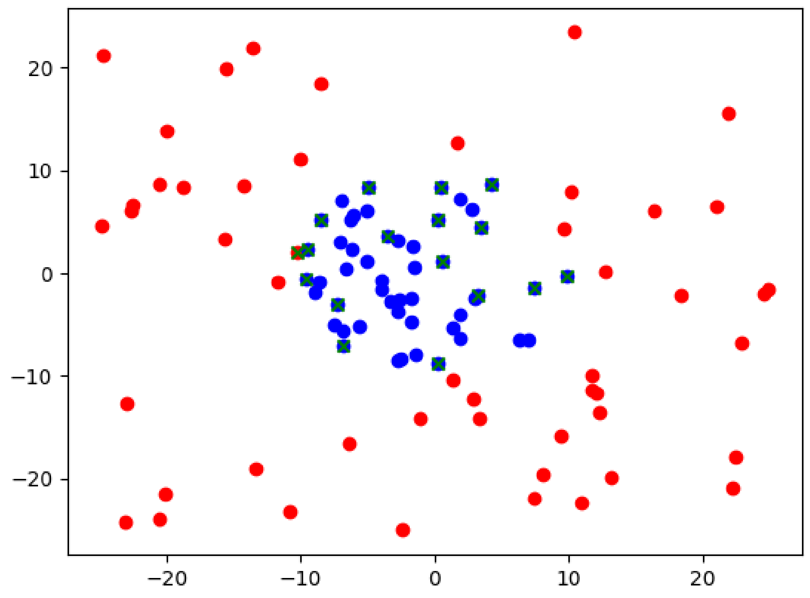
b = -0.10698733753

And the percent identified correctly was 1.0.

For the non-linear case, I first plotted the scatter plot (shown later in results) to attempt to identify what the best kernel would be. It seemed that the radial kernel was the best suited for the data:

$$k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$$

Since we had an object constructed for the SVM, it was a simple matter of sub classing the linear SVN, and overriding the kernel method, as well as the prediction method. Doing so yielded the following figure and results:



Although I was not able to graph the line, we are able to see the support vectors in green x.

The outputs were:

W = [30.602433664956166, -137.63726890979183]

percent correct: 1.0

PART 2 SOFTWARE FAMILIARIZATION

For this stage, I chose to use try and use the WEKA support vector machines. This is called the SMO (and for numerical outputs, SMOreg). Some data transformation had to be done to make the WEKA understand that it is classification problem, by changing the output to string classifications rather than numerical values. (linsep.csv and nonlinsep.csv.)

In choosing the kernel, there were several different kernels. The ones relevant to me were PolyKernel and RBFKernel. Their usage is described by their respective links.

Essentially, the PolyKernel describes:

$$k(x_1, x_2) =< x_1, x_2 >^p$$

Where you specify the polynomial.

RBFKernel is a radial kernel that will allow you to specify the gamma parameter:

$$k(x, y) = e^{-\gamma<x-y,x-y>^2}$$

It is interesting to note that both kernels are sub-classes of the CachedKernel, which means that the kernel output values are cached in a map, (default size of 250007). This must mean that they expect this kernel to be the highest cost function to compute, and needs some way of optimizing its computation.

The output for the linear case can be found in linout.txt. We note that although the weights are quite different from our own calculations, testing against the training set reveals that it's able to predict with 100% accuracy just like ours.

The output for the non-linear case can be found in nonlinout.txt. The output of this kernel provides how many support vectors were found. I first tried this algorithm with gamma=0.5. The output indicated that there were 98 support vectors identified. This seemed weird – as if we are overfitting the problem. I changed the gamma to its default value (0.01) and found that the number of support vector went down to 40. Perhaps this is also why we observe in our own output that while the linear case only had 3 support vectors, our radial kernel yielded much more.

APPLICATION

SVM’s are very elegant mathematical solution to a classification problem. It is rather fast, and is able to handle soft margins as well as non-linear data, which is much better than a linear regression. However, there are some weaknesses. First, it is only able to do classification of two classes. Any more than binary would create a problem. Additionally, the kernel used is up to the designer’s choice…and sometimes one might choose a kernel that isn’t completely mathematically valid – only a ‘guess’. Finally, although soft margin allows an SVM to handle noise, it is relatively sensitive to noise, if there is too much of it.

Some application for an SVM include using the text in an email to classify if the email was or not.

Another interesting application of SVM is Protein classification, for example the Protein secondary structure prediction (PSSP) uses microarray data to predict some structural properties of proteins (helix, coil, or strand structure).