3.2 Support Vector Machine (SVM)

We decided early on to use SVM for two major reason: Its ability to classify data so reliably, and its ability to perform just as strongly in higher dimensions. For classifying data, the main idea behind SVM is to draw a dividing line between two classes of data (called the decision boundary). This line behaves very simple once it is constructed, any data point under it is classified to a certain class, and above to the other class. Before we can use it for classification, we first have to calculate this boundary. As it turns out, this calculation for the line falls into a special type of math known as convex optimization, meaning we aren’t going to be able to use standard algebra to compute it. The exact problem we are looking to solve is known as the dual formulation of the optimization problem. After we solve this optimization, we will have the necessary values (known as Lagrangian Multipliers) to create our decision boundary.

This equation is subject to the following constraints:

As mentioned, we now have the Lagrangian values after solving the equation above. These values will be attached a select few vectors, these are known as support vectors. These are the vectors closest to our decision boundary. We will use these support vectors, and nothing else, to create the boundary. We could throw away the other vectors and still arrive at the same boundary. To create the boundary, we solve the following equations:

In this case, w is a vector representing the weight given to each attribute, and b is the bias. Once we have these values, our line is complete and ready to go. Do note, in the case of data that is not linearly separable (the classes have some overlapping point), SVM doesn’t perform well as it laid out here. For that, the most common approach is to use a kernel to transform the data onto a hyperplane wherein they can be linearly separated. Doing so can produce boundaries that take on very strange shapes when transformed back to the appropriate dimension. We didn’t do much work with huge kernel transformations, so I’ll leave the math behind it out.