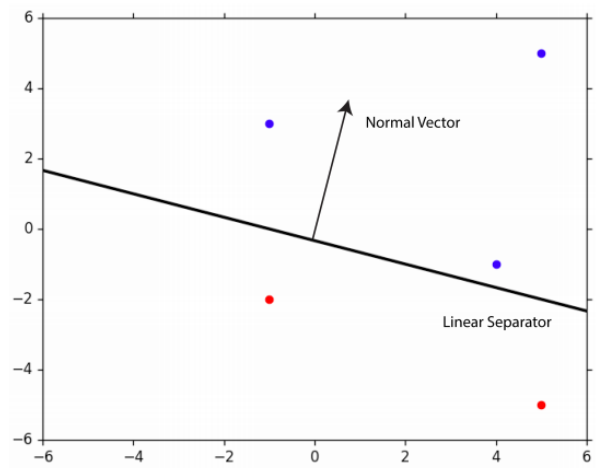
1 Linear Classifiers

A linear classifier is simply a classifier that uses a line to label the data points. Lets say in 2 dimensions we take a line and all points above are positive and all points below are negative. This is a linear classifier.

Interestingly enough, all linear classifiers can be defined in a such manner with a slight modification. Instead of saying above and below (which is ambiguous) we will instead define a line by its normal vector. The normal vector tells us the direction of the line. Why not just use the generic point slope version of a line as definition? Before we answer that question notice that two normal vectors define the same line. The advantage of the normal is that we now say all points on the same side of the line as the normal vector are positively labeled and all points on the other side are negative. This also addresses the fact that one linear separator can lead to two different labelings.



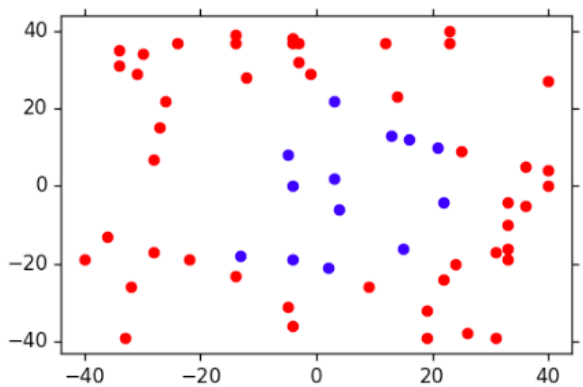
Note that to actually define the line you need the normal vector and one point on the line because the vector only gives the slope. To get the actual location of the line most algorithms use a notion of the offset which relates to how far the line is from the origin.

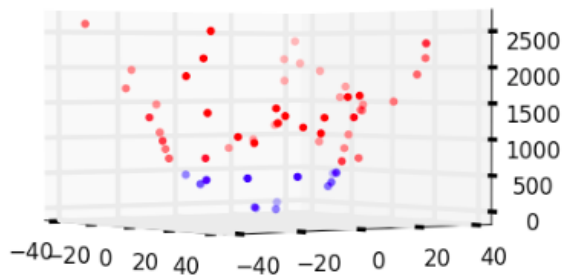
Why learn about linear classifiers? In any field the "why" questions are always the most important and in the field of machine learning this is a natural one. There are a couple clear issues with linear classifiers.

What if the data is not linearly separable?

This is a pretty big problem. If our machine learning algorithms can only classify using linear relationships then we are quite limited. Sure a lot of data may be linearly separable but a lot of data is not.

Who said we have to work with the data strictly as it is inputted. What if we transform the data using a set of functions and then try to linearly classify our transformed data. For example, lets say we have a bunch of data as follows





Here we see an original data set that is radially separable and is simply transformed into a new dataset which is linearly separable. Sure not all data is linearly separable, but if we apply a certain transformation to the data is can be. This is where we see the first true limitation of linear classifiers. All though most data can be expected to obey a certain set of relationships, we obviously cannot cover all transformations without understanding the training data. Of course understanding the data is the point of a machine learning algorithm so in cases where relationships of the data are complex we look to neural networks and other more advanced algorithms.

What if we want more labels than just positive and negative?

Earlier we defined classification as effectively bucketing data points. Using two buckets seems quite limiting. Turns out we can devise schemes that can assign multiple labels using multiple linear classifiers.

One way to do this is to use a binary classification scheme. Lets say I have labels A, B, C, D, E. My first classifier could assign a positive label to data points that are in A or B and negative label to C, D or E. The second classifier could separate our new data-set of points in A + B and decide whether to label them as either A or B.

Another solution could be to create a set of linear classifiers that determine for each individual label if the data point should receive the label or not. So in the A, B, C, D, E case we would have 5 classifiers. Of course this introduces it's own set of issues (what if a point is labeled into two different groups? what if a point is not labeled at all?), but these are reconcilable by making simple augmentations to the classifier.

What this goes to show is that linear classifiers are sufficient and extremely important for this reason. Its hard for algorithms to work with classifier based on complicated functions. It turns out

to be easier to just work with linear classifiers and then simply transform the data. Furthermore, if we want more labels, then we simply use multiple linear classifiers.