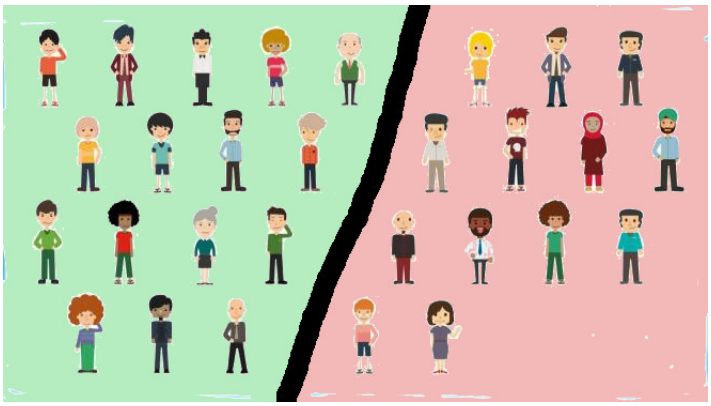
The Perceptron

MATH1900: Machine Learning

Location: http://people.sc.fsu.edu/~jburkardt/classes/ml_2019/perceptron/perceptron.pdf



Let's split the difference!

1 The Dividing Line

Suppose we have the math and verbal SAT scores of all the students who applied for admission to Pitt, and that we also know who got admitted and who didn't. We could use the pair of scores as coordinates. If we used a green dot for admitted students and a red dot for those denied admission, we'd expect to see a rough pattern emerge. Since there are so many applicants, we'd really be interested in determining if there was a simple formula for a line that divides the two sets. If so, we can summarize all the data we have, and perhaps make reasonable predictions about new candidates. The production of a linear formula that splits data into two sections is the goal of the perceptron algorithm.

The perceptron problem

Given n items of m-dimensional data x, some of which belong to a set P, find weights w of a function

$$y = w_0 + \sum_{i=1}^{m} w_i x_i$$

so that

$$\begin{cases} y < 0 \text{ if } x \notin P \\ y > 0 \text{ if } x \in P \end{cases}$$

2 The Perceptron Algorithm

If it not always possible to draw a line (or plane or hyperplane) that correctly separates two sets of data. If we can do this, we say the data is *linearly separable*. For convenience, we will assume for now that our data has this property; however, note that even if our dividing line has a few exceptional points on the wrong side, it would still be a useful rule.

We will expect that if there is one line that separates the data, there are actually many different dividing lines that are possible. For the perceptron algorithm, we will be satisfied with finding *any* such line; later, with support vector machines, we will search for the best possible divider.

While the 2D case is easy to visualize, keep in mind that in 3D, the equivalent separator would be a plane, and in higher dimensions a hyperplane.

To carry out this exercise, retrieve the file `generators_data.txt`, which contains $n = 56$ rows of data about a set of engines: the index, the RPM measurement, the vibration measurement, and a grade (-1 = "defective", 1 = "working"); We will regard our data items as 2-dimensional quqntities storing the RPM and vibration measurements; the value of the grade will be our label, splitting our data into two groups.

Although our data x is 2-dimensional, it will simplify our calculation if we switch to a 3D version, in which the first component is always 1. This will allow us to rewrite the affine formula for y as a truly linear formula.

1. Read the file `generators_data.txt` into `data`.
2. Ignore column 0, extract columns 1, 2, 3 as `rpm`, `vib`, and `grade`.
3. From `rpm` and `vib`, create normalized variables `r` and `v` rescaled to $[0, +1]$;
4. Define the $n \times 3$ array `x` so that column 0 is entirely 1, followed by columns for `r` and `v`.
5. Initialize the 3-dimensional weight vector `w` to 1;
6. Note that our formula is $y = f(r, v) = w * [1, r, v] = w_0 * 1 + w_1 * r + w_1 * v$;
7. Set the "learning rate" `alpha` to 0.01, and carry out the following iteration:

```
1   while ( True )
2
3     changed = False
4
5     for each data item i
6       if 0 < dot product ( w, x[i,:] )
7         f = 1
8       else:
9         f = -1
10      if grade[i] not equal to f
11        changed = True
12        w = w + alpha * ( grade[i] - f ) * x[i,:]
13
14    if ( not changed )
15      break
16
17  w = w / norm ( w )
```

Check your results. It should be the case that `dot ( x[i,:], w )` is positive if the i-th data item has a grade of 1, and negative otherwise. In other words, the perceptron has found weights `w` that classify the data `x`. If you plot the data, you might see that your line has "just barely" separated the data.

```
Perceptron weights
--------------------------------------------------------------------------------
f(r,v) = y = [1,r,v] * w = ___ * 1  +  ___ * r +  ___ * v  (using normalized data)
```