Machine Learning - Clustering

Ashleigh Matherly

There are lots of ways to organize and analyze data. The most used method is linear charts, spreadsheets, or scatter plots. You can visualize this data using graphs or by sorting by value. Computers did not exist too long ago, and clerks had to store data in organized file cabinets. If someone wanted to analyze this data, they would have to read every single file and calculate what they needed manually. Now that we have computers, we can feed them data and have them calculate things for us very quickly.

Machine learning is a relatively new way of processing and storing data to analyze and make predictions. In K-Means Clustering data is stored in "clusters," which has a center that is the mean of all the data points within the cluster, and K is the number of clusters (Tyagi, 2020). If you looked at a scatter plot, you could see if data points look like they are clustering together, and we could determine that the data points in each cluster share similar characteristics.

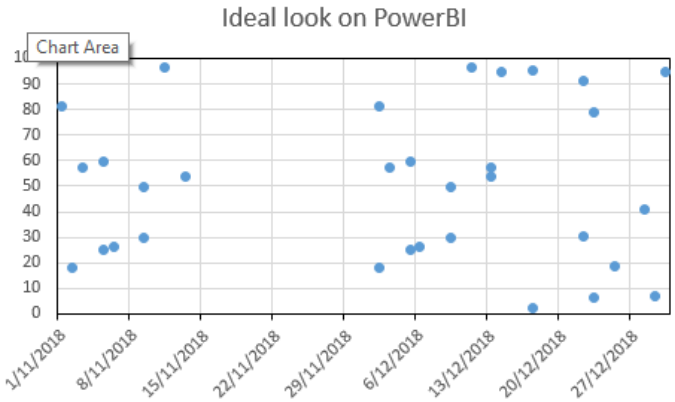

Figure 1. Random Scatter Plot - Source

Looking at the scatter plot in Figure 1, we can make out a potential clustering shown in Figure 2:

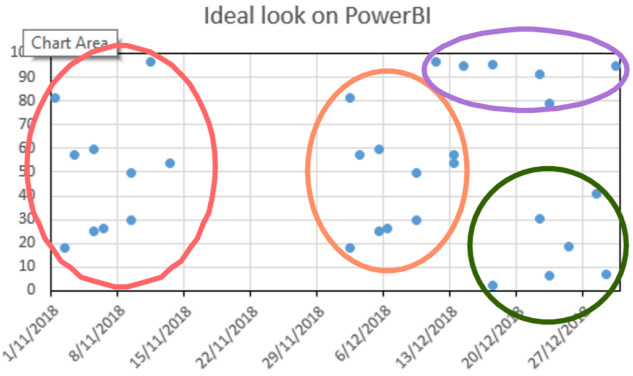

Figure 2. Clustered scatter plot

If we found the area of each ellipse, we could determine the true mean of the cluster, which is needed to make a K-Means cluster. But this isn't the most effective shape to do that, and we would need to make the area between the points as small as possible; so, instead, we should connect the outer points to make a polygonal shape for each cluster:

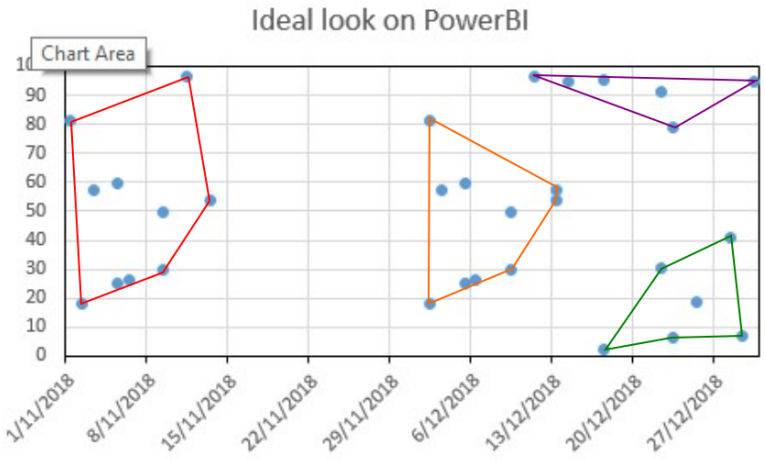

Figure 3. Polygonal clusters

Now let's find an effective way for a computer to quickly find the area of each cluster based on their outer vertices. We can do this by solving it manually first with basic geometry, then calculus to derive something we can translate into a programming language to utilize in a more extensive algorithm.

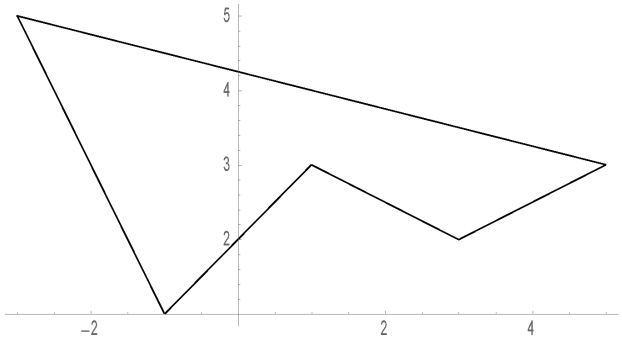

Figure 4. Arbitrary "cluster" shape

Suppose you wanted to calculate the area of the arbitrary polygon in Figure 4. In that case, you could split it up into triangles and rectangles and compute the area using the basic formulas of geometry (Figure 5.)

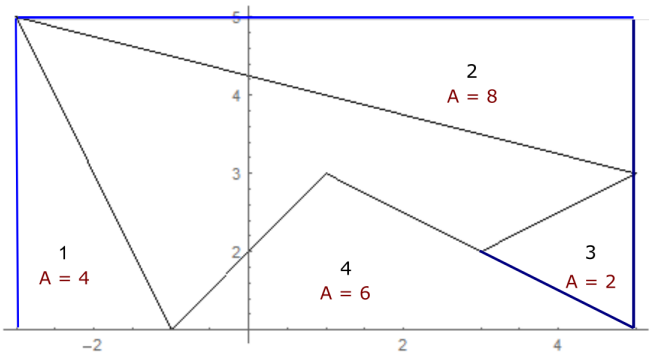

Figure 5. Blue lines were added to make elementary shapes

After making a box around the shape and finding the area of the outer triangles, we can subtract the triangle areas from the area of the box, which gives us 12 units for the polygon's area. This way is easy for humans to do, but not so much for computers, so let's derive a formula so we can write a computer program that can calculate the area of these clusters for us.

First, let's determine the value of the line segment, C, in the vector field $\vec{F}$:

$$\text{Let } \vec{F} = \frac{1}{2}\langle -y, x\rangle.$$

$$\int_C \vec{F} \cdot dr$$

To solve this integral, we need to parameterize the function to put it in terms of t, to give it direction. We can use the known parameterization of a straight line:

r(t):

$$x = x_1(1 - t) + tx_2$$

$$y = y_1(1 - t) + ty_2$$

and

r'(t):

$$-x_1 + x_2$$

$$-y_1 + y_2$$

Which takes a section of the line segment and compresses it down to a segment from:

$$0 \leq t \leq 1$$

Which we will use as our bounds for this integral.

So:

$$\frac{1}{2}\int_0^1 \vec{F} \bullet r'(t)dt$$

$$\frac{1}{2}\int_0^1 \langle y_1(1-t) + ty_2,\ \ x_1(1-t) + tx_2\rangle \cdot \langle -x_1 + x_2,\ \ -y_1 + y_2\rangle dt$$

Using the dot product then simplifying:

$$= \frac{1}{2}\int_0^1 -x_2y_1 + y_2x_1 dt$$

$$= \frac{1}{2}(-tx_2y_1 + tx_1y_2)]_0^1$$

$$= \frac{1}{2}\left(x_1y_2 + x_2y_1\right)$$

Now that we have a formula to calculate a line integral of a straight line from two points, we can use Green's Theorem to calculate the area of any polygon of a vector field.

Green's Theorem states that if the partial derivative with respect to y of the P component of the vector field minus the partial derivative with respect to x of the Q component of the vector field is equal to 1, we can simply take the double integral over the area of the shape.

To demonstrate this, let's take Green's Theorem, defined as:

$$\oint Pdx + Qdy = \iint (Q_x - P_y)dA$$

Where dA is the area of the region.

Our vector field from above is defined as:

$$\vec{F} = \frac{1}{2}\langle -y, x\rangle$$

$$P = -\frac{1}{2}y,\ P_y = -\frac{1}{2}$$

$$Q = \frac{1}{2}x,\ Q_x = \frac{1}{2}$$

So:

$$\iint \frac{1}{2} - \left(-\frac{1}{2}\right) dA = \iint 1 dA$$

This means our vector field doesn't affect the area of the region because it evaluates to 1. Now to find the area of the polygon, let's take the area of each line integral using the derivation from earlier and add them all up, represented by the series:

$$\frac{1}{2}\sum_{i=1}^{n}\left(x_i y_{i+1} - x_{i+1} y_i\right)$$

Now that we have a straightforward way to find the area of polygons, we can translate this into a programming language to have a computer calculate it for us. I wrote a program that takes known vertices of any polygon, which uses the shortcut derived above to sum the line integrals of each line segment to return the area of the region, all without computing any integrals. The program asks for the known vertices of the region and returns the area. This program is just a small portion of what a machine learning algorithm would use, but it is the key to setting up an ideal cluster. Although some human calculations are required in data science and machine learning; we can now process data more and more efficiently as technology progresses.

```
import java.util.Scanner;

// This program asks the user for the number of sides of a polygon, then
prompts them to input each corner point.
// It will calculate the area of any polygon with more than three sides.
public class PolygonCalculator {
   public static void main(String[] args) {

       Scanner scanner = new Scanner(System.in);
       System.out.println("Number of sides:");
       int sides = scanner.nextInt();

       double[] xCoords = new double[sides + 1];
       double[] yCoords = new double[sides + 1];

       //Prompting the user for the coordinates of each point
       for (int i = 0; i < sides; i++) {
           System.out.println("x sub " + (i + 1) + ":");
           xCoords[i] = scanner.nextInt();
           System.out.println("y sub " + (i + 1) + ":");
           yCoords[i] = scanner.nextInt();
       }

       xCoords[sides] = xCoords[0];
```

```
        yCoords[sides] = yCoords[0];

        System.out.println("Area: " + findArea(xCoords, yCoords));
    }

    public static double findArea(double[] x, double[] y) {
        double area = 0;
        for (int i = 0; i < x.length - 1; i++) {
            // the formula derived from the sum in the document
            area += (x[i] * y[i + 1] - x[i + 1] * y[i]);
        }
        // if area is negative, flip the sign
        if (area < 0) return area / -2;
        else return area / 2;
    }
}
```

Works Cited

Tyagi, Neelam. 2021. “What Is K-Means Clustering in Machine Learning?” *Analytics Steps*, https://www.analyticssteps.com/blogs/what-k-means-clustering-machine-learning.