The Theory of Pattern Analysis

Athan Johnson

November 2025

Abstract

I theorize that contrary to popular belief, P is equal to NP. I therefore believe that it is possible to find a polynomial time algorithm that can find the solution to nondeterministic polynomial problems. I also believe that there are overarching patterns in these problems such that there is potential for an algorithm to be made to generate the corresponding solution algorithm for any NP problem. However, such an algorithm I think would likely find these specific solution algorithms in exponential time.

1 Introduction

I will define a pattern as any sequence of points in space. The sequence could be infinite, such as an unending string of 0101... or in multidimensional space such as two points on a graph.

Let us consider a common pattern seen in calculus classes: a series of points on a graph. If we randomly generate a finite number of points on this graph, there exist infinitely many polynomials that pass through these points. In fact, there are infinitely many degrees of polynomials that exist that can pass through these points. Even infinitely many points possess infinitely many polynomials of infinite degree which can represent the problem. The challenge is this: is there a way to represent all of these polynomials? A formula which could be given the points, and output the corresponding representation of these infinite polynomials which solve it?

Consider $(0, 0)$ as our starting point. How can you represent all polynomials that pass through this single point? According to Google, $p(x) = A_n x^n + A_{n-1}x^{n-1} + ... + A_1 x^1$ is the formula to represent all polynomials that pass through $(0, 0)$. Changing the y-value here is simple, it adds a constant to the end. The formula for all points that pass through $(0, 1)$ is $p(x) = A_n x^n + A_{n-1}x^{n-1} + ... + A_1 x^1 + 1$.

Changing the x-values is only a little trickier. The polynomials passing through the point $(1, 0)$ is represented by $p(x) = A_n(x-1)^n + A_{n-1}(x-1)^{n-1} + ...+A_1(x-1)^1$. So now we have a simple formula to find all polynomials that pass through a given point (a, b). For any point (a, b) the formula for all polynomials that pass through it is $p(x) = A_n(x-a)^n + A_{n-1}(x-a)^{n-1} + ... + A_1(x-a)^1 + b$.

Can this be generalized further? For example, if we have a set of data points in space can we use a formula to find all possible polynomials that pass through those points? Can we find a formula that finds all possible *expressions* that pass through those points? Consider two points in space, (a, b) and (c, d). For point (a, b) all possible formulas that pass through it are, as we established, $p(x) = A_n(x-a)^n + A_{n-1}(x-a)^{n-1} + ... + A_1(x-a)^1 + b$. And for point (c, d) all points that pass through it are $q(x) = B_n(x-c)^n + B_{n-1}(x-c)^{n-1} + ... + B_1(x-c)^1 + d$. Considering the sets of $p(x)$ and $q(x)$ we know that there are polynomials that fall outside both equations, fall within both equations, and some that fall within one and not the other (assuming a, b, c, and d are all different). How can we express the set of formulas that all fall within both equations?

2 Proving that P = NP

Assuming that we have found a formulaic way of representing polynomials, we should then be able to convert our two dimensional space into a tape, one readable by a Turing Machine. This means that our formula for finding polynomials can itself be converted into a Turing Machine. A formula such as the one I imagine is possible should simply be able to take in the points given to it, and output for us all the possible polynomials which pass through these points. This implies the existence of what will hopefully be a linear time Turning Machine capable of identifying polynomial patterns within a given tape, and from there we need a way to convert an NP complete problem into our polynomial, or the Turing Machine or something.

This paper still needs quite a bit of work before it's complete, but time will tell if I am able to figure out these various holes and finish it.