# AEON
# A DUMMY APPROACH TO P=NP THROUGH SAT

CHARLES DANA, DISCRETE MATHEMATICIAN, H24

Abstract. Let $\mathbb{L}$ be a subset of $\{0,1\}^{\mathbb{N}}$, there exists four operators namely *AND OR ELSE NOT*, that suffice to describe a SAT instance $\phi$ that follows:

$$\phi(X) = 1 \Leftrightarrow X \in \mathbb{L}$$

It can be proven by recursion on the alphabet defined by $\mathbb{L}$. This only applies if $\mathbb{L}$ is defined on a finite number of variables, ie: $\mathbb{L} = L \times \{0,1\}^{\mathbb{N}}$ where $L \subset \{0,1\}^n$. Furthermore, it will generate what we call dummy variables, $|l| > n$, which complexify the restitution.

Contents

## 1. What is an AOEN?

You have a list of statements in the form $\{T/F\} \times \{+,-\}^n$. The question you are asking is what's the easiest way to describe these statements (T/F) out of the context (+,-). For what I would answer recursively. It suffice to be capable of reducing an AEON to a SAT instance, if everything contained from within the AEON is to be expressed as a SAT instance.

1.1. **Prerequisites.** What is a SAT instance? $(\mathcal{M}_{m,n+d}(-1,0,1),n)$, which is to say a collection of $m$ clause on $n$ dependent variables, with $d$ dummy variables.

1.2. **Claim.** There exists a unique solution set on $\{0,1\}^n$ associated with any given SAT instance.

1.3. **Claim.** There is a relationship of equivalence, built on the computation of the dummy variables that encode the truth statement of every assertion.

1.4. **Claim.** Any Given AOEN is a SAT Instance, with dummy variables.

1.5. **Claim.** If $P \neq NP$, there are AOEN that will not be computable as solvers will have exponential complexity on the number of dummy variables.

1.6. **Claim.** If $P = NP$, there is an algorithm that reduces strictly the number of clause of an equivalent SAT instance, in its description of what it is true, in polynomial time.

1.7. **Claim.** There exists promising prototypes that will solves and reduce cryptography to a simplistic problem. But in the meantimes fully encrypted, publicly shared information will be made possible.

## 2. What is the purpose of an AOEN

Suppose you have a 0-1 that is both deterministic, and reliable. You can, using an AOEN associate the result of your boolean, to a SAT variables. Which means, being apart of an AOEN. But Because by nature AOEN is SAT, there will be a logic instance, that will fit perfectly your training data.

Suppose you have a model with a signal $t/f$ that will match or not the language $\mathbb{L}$ solution set.

| $Signal$ | $t$ | $f$ |
|---|---|---|
| $T$ | $tT$ | $fT$ |
| $F$ | $tF$ | $fF$ |

You wish to diagonalise this matrix, ie: Maximizing $(tT + fF)^2 - (fT + tF)^2$. Remember $(T/F)$ is your training signal. So you compute your $\mathcal{M}_2(\mathbb{N})$ scoring matrix. And you check for $fT$, so you subcategorize on one of these subsets $B$ that contains them, and use the trick.

2.1. **Theorem of lesser evil.**

$$\mathcal{A}_1 = (t/f)$$

$$\mathcal{A}_2 = OR(\mathcal{A}_1, (AND(NOT(\mathcal{A}_1), B)))$$

$$\mathcal{A}_2 \geq \mathcal{A}_1$$

Until there are no more fT. You can force isolate the $tF$.

## 3. Is there a mathematical proof?

Yes, and it's a good one. Assume your variables are the collection of signals from ML, your expert gives your the best combination: You fit what you know for the truth values of the variables, and a computer gives you an insight on if the question you are asking is True or False. Proof by python:

```
def reduce(SAT):
"""
Returns a NEW instance
with the exact same Truth values set,
but a lesser or equal number of clauses.
"""
    ...
    return NEW

def improve(AOEN):
"""
Returns self or (not self and should be self)
with a greater or equal fit score than the previous one
"""
   ...
   return CLEVER

def transpose(AOEN):
"""
SAT with the number of variables n and the dummy variables associated.
"""
   ...
   return SAT
```

## 4. Proofs.

4.1. **Proof of the prerequisite.** The prerequisite proof is that if you read the $\mathcal{M}_{m,n+d}(\{-1,0,1\})$ matrix you obtain a unique sat instance of $m$ clause. Both sets are in bijection, and finite on fixed m,n,d. The idea is that you only have info

up to $n$ so replacing the statements generate a unique instance on the $d$ dummy variables. A fast UNSAT Solver will answer the 0-1.

You might have a logic instance as an answer.

4.2. **Proof of Claim.** Do not fear there are solutions. Assume you can compute till infinity on a finite world ie, no cap on the constant. Any SAT instance with $m$ variables is nothing but a subset $L$ of solutions in $\{0,1\}^n$ that do not care for higher variables. Thus $\phi = L \times \{0,1\}^{\mathbb{N}}$.

4.3. **Proof of Claim.** Assume that there is a signal that should be set to true, but no remaining dummy variables allows it, without generating a contradiction. It means that even if you knew the key, you would get a False value, thus UNSAT is preserved. What about SAT, if it is false and should be set to true. Interesting case and the theorem of lesser evil allows us to brute force a solution, that might be reduce to fit and depict, the terms, like overfitting.

4.4. **Proof of Claim.** Assume your AOEN is its basic SAT form, ie: list of lists of integers. Is there a universal way to compile the AOEN? The idea is to use an AND Statements on each output.

$$\phi_A \wedge \phi_O \wedge \phi_E \wedge \phi_N$$

The compile the list of SAT instances in *And* it suffice to append them one to the other. The compile of the list of SAT instances in $Or$, needs a chain of dummy variable, given to $problem_1 \vee d$ and $-d \vee problem_2 \vee (d+1)$ and $-(d+1) \vee problem_3 \vee (d+2)$, which introduces a number of dummy variables equal to the number of subsat instances in the $Or$. *Else* usually introduces a single SAT instance, and can be replaced by the output of a neural network, that can be translated into SAT. $Not$ exists, and generates $\log(m)$ dummy variables.

4.5. **Proof of Claim.** If $P \neq NP$ some instances will just be out of reach of current solvers, because you need OR and NOT Statements to improve an existing Solver. But it will be some relatively easy constant added per layers. Make sure that your dummy variables remain independant.

4.6. **Proof of Claim.** There is a bijection on a subset of $\mathbb{R}^{n+1}$ and any clause of a SAT instance $\phi$. Which means you can reduce the number of clauses by at least one with a reliable probability. (using continuous analysis).

4.7. **Proof of Claim.** We are in the process of constituting Algorithmes, a venture driven by a research lab on SAT instances. As of 2024-02-07, we have a universal boolean classification algorithm in polynomial time. As well as a probabilistic SAT reduction algorithm that solves instances.