

Introduction to Decision Making Algorithms

Jeanna Matthews
Clarkson University

November 16, 2018 Conference on Trade Secrets and Algorithmic Systems

Decision Making Algorithms

- Big decisions about the lives of individuals are being made in a partnership between human decision-makers and computer systems.
- Fundamentally changing the landscape of our societal decision-making processes
 - Criminal justice, hiring, housing, credit, news amplification, elections, ...
- In an environment dominated by trade secrecy, what will be the incentives for iterative improvement/debugging? Fairness? Respect of fundamental societal principles?



Algorithm

- Unambiguous specification of how to accomplish a task
- Step-by-step instructions
- Recipe



How implemented?

- How is algorithm implemented or executed?
 - Humans follow directions?
 - Software? Hardware? Partnerships?
- The more complex the algorithm, the more software or hardware is needed to implement it
 - Automated System
 - Automated Decision Making



LRmixStudio - example

Help

Sample Files Reference Files Profile Summary Analysis Sensitivity Analysis Non-contributor Test Reports About

Active	Sample	Source File
<input checked="" type="checkbox"/>	Rep1	sample.csv
<input checked="" type="checkbox"/>	Rep2	sample.csv
<input checked="" type="checkbox"/>	Rep3	sample.csv

Case Number example Restore session from Log Restart Add replicate Load from file...

Locus	Rep1	Rep2	Rep3
D10S1248	13 14 16	13 14 15 16	13 14 15 16
VWA	15 17 18 19	15 16 17 18 19	15 16 17 18 19
D16S539	11 13	10 11 12 13	10 11 12 13
D2S1338	17 18 19	17 18	17 18 19
D8S1379	10 12 14 15	10 12 14 15	10 11 12 14 15
D21S11	28 29 30 33.2	28 29 30 32.2 33.2	28 30 33.2
D18S51	14 15 17	10 14 15 17	14 15 18
D22S1045	11 12 15 16	12 15 16	11 12 15 16
D19S433	13 14	13 14	13 14
TH01	6 7 8 9.3	7 8 9 9.3	7 8 9.3
FGA	18 21 24	18 19 22 23 24	18 21 22 23 24
D2S441	10 11	10	10 11 13
D3S1358	16 17	15 16 17	15 16
D1S1656	12 15 16 18	11 12 15 16	12 16
D12S391	15 17 19 20 23	15 17 19 20	15 17 19 20

Where did the specification come from?

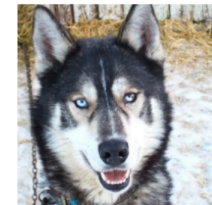
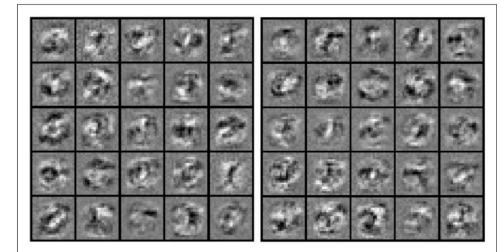
- System designer/developer
- Rule-based systems learned from domain experts
- Learned from data
 - Looking for patterns in data/ “facts” about the world
 - Often still fundamentally learned from humans: manual classification of training data or past data that reflects human decisions



Machine Learning Systems

- Typical process of classification
 - Manual labeling
 - Learning from past “successes”
- Impact of training data
- Dogs in the snow
- Learn from but be careful not to reproduce the past

5	0	3	7	5	4	6	4	4	6
4	6	1	0	7	6	3	5	3	9
1	5	1	8	6	6	5	0	7	3
7	4	8	4	7	6	6	5	6	1
0	0	7	6	0	2	9	7	5	0
6	2	6	9	4	6	6	8	6	6
1	0	2	5	4	6	7	4	0	2
7	0	4	6	7	3	2	7	7	7
1	9	3	0	3	2	8	9	2	2
9	8	0	0	1	1	4	0	6	1



(a) Husky classified as wolf



(b) Explanation

Figures from “How the Machine ‘Thinks!’ Understanding Opacity in Machine Learning Algorithms”, B
and “‘Why Should I Trust You?’: Explaining the Predictions of Any Classifier”, Ribeiro et al.

Defining correctness

- How is correctness defined?
 - Does the implementation faithfully follow the specification?
 - What if the specification is incorrect/incomplete?
- Other metrics?
 - Accuracy of prediction?
 - Impact on society?



Correctness?

- Is it correct?
 - For a particular case?
 - For all cases?
- Are people capable of even determining whether it is correct?
 - Which people?
 - Systems that are too complex to be manually verified.
- Is it understandable/explainable?
 - To which people?



Bugs

- Complex systems and automated systems have bugs
 - Anyone who or uses them knows this!
- They cannot be correct without transparency and iterative improvement

The 5 Stages of Debugging

At some point in each of our lives, we must face errors in our code. Debugging is a natural healing process to help us through these times. It is important to recognize these common stages and realize that debugging will eventually come to an end.



Denial

This stage is often characterized by such phrases as "What? That's impossible," or "I know this is right." A strong sign of denial is recompiling without changing any code, "just in case."



Bargaining/Self-Blame

Several programming errors are uncovered and the programmer feels stupid and guilty for having made them. Bargaining is common: "If I fix this, will you please compile?" Also, "I only have 14 errors to go!"



Anger

Cryptic error messages send the programmer into a rage. This stage is accompanied by an hours-long and profanity-filled diatribe about the limitations of the language directed at whomever will listen.



Depression

Following the outburst, the programmer becomes aware that hours have gone by unproductively and there is still no solution in sight. The programmer becomes listless. Posture often deteriorates.



Acceptance

The programmer finally accepts the situation, declares the bug a "feature", and goes to play some Quake.

Legal Protections

- Intellectual property claims used to keep away legitimate concerns about correctness
- DeWitt clauses in terms of service documents used to stifle reporting of problems
- Anti-reverse engineering used to prevent thorough third-party testing



Incentive for debugging?

- In this environment, essential to ask “what is the incentive for debugging and iterative improvement?”
- Doomed to run society on buggy systems if we don't enable iterative improvement



Interests of developers vs. deciders vs. those decided about

- Interests of system developers or system customers are often different than interests of those being decided about
 - Rare cases that matter to individuals
 - Often boils down to efficiency or reduced risk for the decision maker versus protection for the individual
 - Invest some of savings in robust investigation of errors
 - Tax on deciders – but that is not new!
- Criminal justice applications perfect example
 - Interests of developers? Interests of deciders?
 - Rights of defendants? Rights of society?
 - Debugging left to individual defense teams
 - What might be changed out from under us in the process of careless automation without incentives for transparency and iterative improvement

Algorithm = specification

- Specification makes decisions auditable and questionable
- What good is specification if we lock it up in a black-box automated system and don't allow auditing, questioning?



- What types of review might attorneys and judges seek in understanding software-based/computer-based evidence?
- Why law and public policy require disclosure of these materials to the public and independent experts?



Opening the Black Box: Defendants' Rights to Confront Forensic Software

Despite this country's commitment to fair and open trials, people are being convicted on the basis of secret computer code. When neither the public nor the accused is allowed to look at how the software operates, it undermines the legitimacy of the judicial system and can send innocent people to prison or to their execution.

Forensic software is used in the criminal justice context to make assertions about the presence and nature of DNA, to deploy police resources to certain areas, or to guide bail and sentencing determinations.

Software, however, is far from impartial or infallible. It is simply a set of instructions to a computer, programmed by fallible humans or trained on flawed historical data sets. Errors both intentional and unintentional are routinely discovered when independent experts are able to analyze these tools.

This article provides advice for understanding and confronting software-based evidence in criminal

prosecutions. The advice falls primarily into two categories. First, from a computer science perspective, the article describes different types of review that attorneys and judges might seek in understanding software-based evidence. Second, from a legal perspective, the article explains why law and public policy require disclosure to the public and independent experts, such as those working with the defense, of the relevant software source code and other software development records, including any training data sets.

In particular, the article explains why courts must reject the idea that a vendor's purported commercial interest in trade secrets should override the rights of a defendant who is at risk of imprisonment or death, or the public's right to the open and fair administration of justice.

I. What Information Do Defense Experts Need to Evaluate Forensic Software?

A. Source Code and Executables: What Does It Mean to Evaluate Software?

Generally, software does what it is programmed to do, including any bugs and biases programmed into it by its creators. Everyone has experienced glitchy software, and everyone has been frustrated when software does not behave the way they expect it to or does not give them the options they need. Software often evolves over time, removing bugs and adding or

BY STEPHANIE J. LACAMBRA, JEANNA MATTHEWS,
AND KIT WALSH

Executables

LRmixStudio - example

Help

Sample Files Reference Files Profile Summary Analysis Sensitivity Analysis Non-contributor Test Reports About

Active	Sample	Source File
<input checked="" type="checkbox"/>	Rep1	sample.csv
<input checked="" type="checkbox"/>	Rep2	sample.csv
<input checked="" type="checkbox"/>	Rep3	sample.csv

Case Number

Locus	Rep1	Rep2	Rep3
D10S1248	13 14 16	13 14 15 16	13 14 15 16
VWA	15 17 18 19	15 16 17 18 19	15 16 17 18 19
D16S539	11 13	10 11 12 13	10 11 12 13
D2S1338	17 18 19	17 18	17 18 19
D8S1179	10 12 14 15	10 12 14 15	10 11 12 14 15
D21S11	28 29 30 33.2	28 29 30 32.2 33.2	28 30 33.2
D18S51	14 15 17	10 14 15 17	14 15 18
D22S1045	11 12 15 16	12 15 16	11 12 15 16
D19S433	13 14	13 14	13 14
TH01	6 7 8 9.3	7 8 9 9.3	7 8 9.3
FGA	18 21 24	18 19 22 23 24	18 21 22 23 24
D2S441	10 11	10	10 11 11.3
D3S1358	16 17	15 16 17	15 16
D15S1656	12 15 16 18	11 12 15 16	12 16
D12S391	15 17 19 20 23	15 17 19 20	15 17 19 20

Source Code

```
nyc-dna-software/Comparison x
GitHub, Inc. [US] | https://github.com/propublica/nyc-dna-software/blob/master/FST.Common/Comparison.cs
// This function checks for the total frequencies according to races and removes the alleles from calculation
// If the sum of frequencies are greater than 0.97.
// </summary>
public void CheckFrequencyForRemoval(DataTable dtFrequencies)
{
    // If our db connection isn't initialized, do it. Then, get all the ethnicities (races)
    myDb = myDb ?? new Database();
    DataTable raceTable = myDb.GetAllEthnics();
    int intsr = 0;
    string[] srem = new string[comparisonLoci.Count];

    // We go through all the comparison loci and check whether the sum of the frequencies for that locus is greater than 0.97.
    // If it is, we remove the locus. Frequencies are only used for the alleles in the evidence replicates.
    for (int i = 0; i < comparisonLoci.Count; i++)
    {
        bool blRemove = false;
        // Get a CSV list of alleles for all the replicates at a locus
        IEnumerable<string> unknownPair = EvidenceAllelesAtLocus(evidenceAlleles[comparisonLoci[i]]);
        // Check if the frequency is greater than 0.97 for any of the races. Frequencies are values for an allele at a locus for a certain race
        foreach (DataRow eachRow in raceTable.Rows)
        {
            string raceName = eachRow.Field<string>("EthnicName");
            float freqSum = GetFrequencySum(unknownPair, comparisonLoci[i], raceName, dtFrequencies);

            if (freqSum >= 0.97)
            {
                blRemove = true;
                break;
            }
        }
        if (blRemove)
        {
            srem[intsr] = comparisonLoci[i];
        }
    }
}
```

Other parts of specification

- Information from the development process
 - Design documents, testing plans and results
- Experience with deployed software
 - Bug reports, change logs



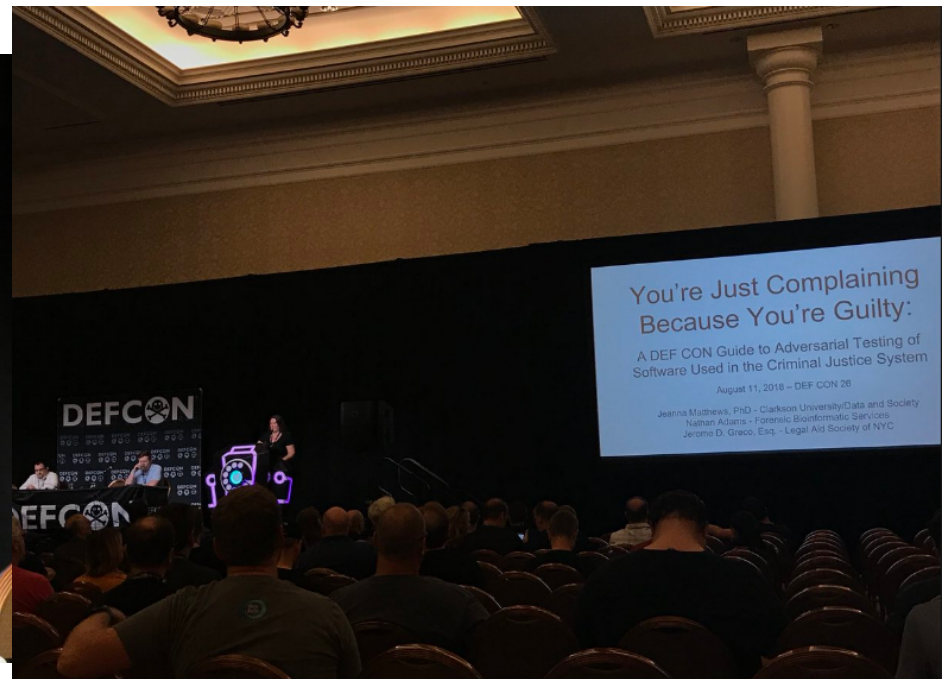
Brown Institute Magic Grant: Decoding Differences in Forensic DNA Software



Methods

- Independent, third-party, adversarial testing and review
 - Automated testing harnesses
 - Common file formats and settings
 - Source code analysis
- Recommendations
 - Clear advise for judges, defense attorneys, journalists
 - Sample requirements for software systems, targeting the procurement phase

DEFCON Talk: “You’re Just Complaining Because You’re Guilty”






- Upcoming article in AI Magazine and talk at 2019 AI For Good

- “Patterns and Anti-Patterns, Principles and Pitfalls: Accountability and Transparency in AI “

- 10 Common Anti-Patterns

1. Learn from the Past Without Remembering the Context
2. Learning from Humans Without Remembering Human Bias and the Possibility of Malicious Training
3. Using Data You Have Rather than the Data You Need
4. Failing to Measure the Social Impact of Deployed Systems
5.

Final Words

- Introduction to decision making algorithms
 - Human decision making vs. automated decision making
 - Specificity, Repeatability, Complexity
 - Importance of Incentivizing Iterative Improvement
 - Protection for Individuals and the Public Good (Not Just Efficiency and Reduced Risk for Deciders)
- 

Thank you!

jnm@clarkson.edu

<http://www.clarkson.edu/~jnm>

[@jeanna_matthews](#)

