When Algorithms Decide

Fairness, Bias, and the Ethics of Automated Decisions

Brendan Shea, PhD

Programming and Problem Solving • Rochester Community and Technical College

Introduction

Should a computer program decide whether you get a job interview? Whether you're approved for a loan? Whether you're released from jail before your trial? These questions might sound like science fiction, but they describe reality. **Algorithms** now make or influence countless decisions that profoundly affect people's lives—often without those people knowing an algorithm was involved.

This case study explores what algorithms are, how they've come to wield such power, and most importantly, what it means for an algorithm to be "fair." As you'll discover, fairness is far more complicated than it first appears. Different definitions of fairness can contradict each other, forcing us to make difficult trade-offs. These aren't just technical problems for computer scientists—they're ethical questions that affect everyone.

The Stakes Are Real

In the United States alone, algorithms influence decisions about who gets hired at major companies, who receives loans, who sees which political advertisements, what news stories appear in your feed, how long criminal sentences should be, and whether patients receive certain medical treatments. Learning to think critically about algorithmic decision-making is no longer optional—it's essential.

1 What Is an Algorithm?

An **algorithm** is simply a precise sequence of steps for solving a problem or accomplishing a task. You follow algorithms every day: a recipe is an algorithm for cooking a dish; directions to a friend's house are an algorithm for navigation; the method you learned for long division is an algorithm for arithmetic.

What makes computer algorithms special is that they can be executed automatically, at tremendous speed, on massive amounts of data. A human loan officer might review fifty applications per day; an algorithm can evaluate millions per hour. This scale transforms algorithms from mere tools into systems that shape society.

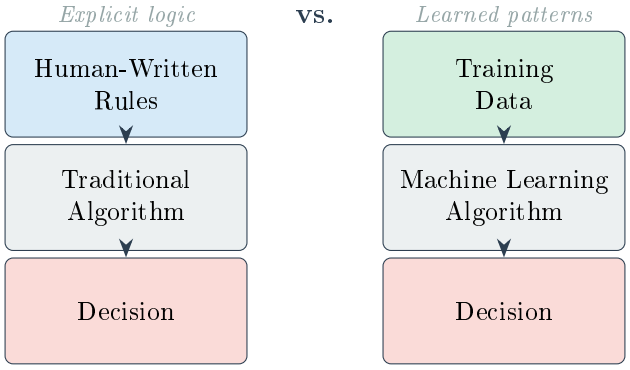
A Simple Decision Algorithm

```
// Should we approve this loan application?
public boolean approveLoan(Applicant app) {
    int score = 0;

    // Award points for positive factors
    if (app.creditScore > 700) score += 30;
    if (app.yearsEmployed > 2) score += 20;
    if (app.income > 50000) score += 25;
    if (app.hasCollateral) score += 15;

    // Subtract points for risk factors
    if (app.missedPayments > 0) score -= 20;
    if (app.bankruptcyHistory) score -= 40;

    return score >= 50;  // Approve if score meets threshold
}
```

This simple algorithm illustrates key features of algorithmic decision-making. It takes **inputs** (data about the applicant), applies **rules** (the if-statements that award or subtract points), and produces an **output** (approve or deny). The rules encode assumptions about what makes someone creditworthy. But are these assumptions correct? Are they fair?

1.1 From Rules to Learning

The loan algorithm above uses rules that humans explicitly wrote. But increasingly, algorithms *learn* their rules from data. **Machine learning** algorithms are given many examples (thousands of past loan applications and whether they defaulted) and discover patterns themselves. This can uncover subtle relationships humans might miss—but it can also learn patterns we wouldn't endorse if we knew about them.



2 The Problem of Bias

In 2014, Amazon built a machine learning system to review résumés and identify promising job candidates. The algorithm was trained on ten years of hiring data—résumés the company had received and which candidates were ultimately hired. The system learned to predict which new applicants would be successful.

There was just one problem: it learned to penalize résumés that included the word "women's" (as in "women's chess club captain") and to downgrade graduates of all-women's colleges. Why? Because the training data reflected a decade of hiring in the male-dominated tech industry. The algorithm learned that being male correlated with being hired—not because men were actually better candidates, but because of historical **bias** in human decisions.

Case Study: COMPAS and Criminal Justice

COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is an algorithm used across the United States to predict whether a defendant will commit future crimes. Judges use these "risk scores" when deciding bail, sentencing, and parole.

In 2016, the investigative news organization ProPublica analyzed COMPAS predictions for over 7,000 defendants in Florida. They found that Black defendants were almost twice as likely as white defendants to be incorrectly labeled as high-risk (meaning they were predicted to reoffend but didn't). White defendants, meanwhile, were more likely to be incorrectly labeled as low-risk (meaning they were predicted not to reoffend but did).

The company that makes COMPAS, Northpointe (now Equivant), disputed these findings. They argued their algorithm was fair by a different measure: among defendants who received the same risk score, Black and white defendants reoffended at similar rates.

Both claims were mathematically true. How could the algorithm be simultaneously fair and unfair?

The COMPAS controversy reveals something profound: **algorithmic fairness** isn't a single concept with a single definition. There are multiple, mathematically incompatible ways to define what "fair" means.

3 Competing Definitions of Fairness

Philosophers, mathematicians, and computer scientists have proposed many definitions of algorithmic fairness. Here are three major approaches:

3.1 Demographic Parity

Demographic parity (also called statistical parity) requires that the algorithm produce positive outcomes at equal rates across different groups. If 30% of male applicants are hired, then 30% of female applicants should be hired too.

Testing for Demographic Parity

```
// Check if hiring rates are similar across groups
double maleHireRate = (double) malesHired / malesApplied;
double femaleHireRate = (double) femalesHired / femalesApplied;

boolean hasParity = Math.abs(maleHireRate - femaleHireRate) < 0.05;
// True if rates are within 5% of each other
```

The appeal is obvious: equal outcomes suggest equal treatment. But critics argue this definition has problems. What if the applicant pools differ in relevant qualifications? Forcing equal outcomes might mean hiring less-qualified candidates from one group or rejecting more-qualified candidates from another.

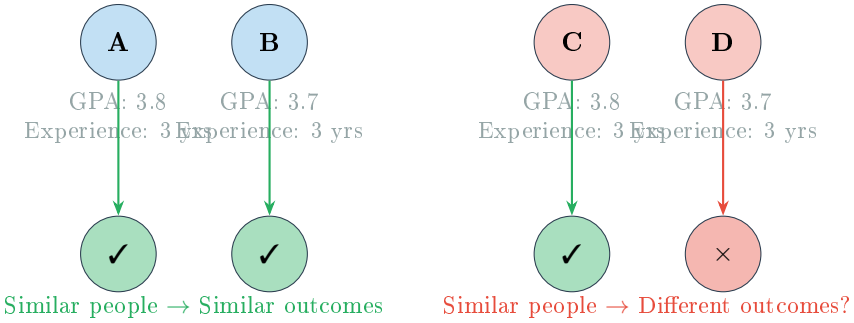
3.2 Equal Accuracy

Equal accuracy (sometimes called equalized odds) requires that the algorithm make errors at equal rates across groups. If the algorithm incorrectly rejects 5% of qualified male applicants, it should also incorrectly reject 5% of qualified female applicants.

This focuses on the algorithm's *mistakes* rather than its overall outcomes. The intuition is that people shouldn't face a higher burden of proof based on their group membership. A qualified applicant deserves fair consideration regardless of demographic category.

3.3 Individual Fairness

Individual fairness takes a different approach entirely: similar individuals should be treated similarly. Rather than comparing groups, it asks whether two people with nearly identical qualifications receive nearly identical treatment.



The challenge with individual fairness is defining "similar." Which characteristics matter? Is someone with a 3.8 GPA and 2 years of experience "similar" to someone with a 3.5 GPA and 4 years of experience?

4 The Impossibility of Perfect Fairness

Here's the uncomfortable truth that emerged from the COMPAS debate: mathematically, you often *cannot* satisfy multiple fairness definitions simultaneously. This isn't a failure of clever engineering—it's a proven mathematical impossibility.

The Fairness Impossibility Theorems

In 2016, researchers including Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan proved that except in special cases, no algorithm can simultaneously achieve:

1. **Calibration**: Among people who receive the same risk score, actual outcomes should be similar across groups.
2. **Equal false positive rates**: The rate of incorrectly flagging someone as high-risk should be equal across groups.
3. **Equal false negative rates**: The rate of incorrectly missing someone who is high-risk should be equal across groups.

The only exception is when base rates are identical across groups—for instance, if men and women actually defaulted on loans at exactly the same rate. In the real world, base rates

often differ (for complex historical and social reasons), making perfect fairness by all measures impossible.

This impossibility forces us to make choices. Which type of fairness matters most in a given context? The answer depends on values, not just mathematics. In criminal justice, incorrectly labeling someone as high-risk (a false positive) means they might stay in jail awaiting trial. Incorrectly labeling someone as low-risk (a false negative) means they might commit another crime. Which error is worse? That's an ethical question, not a technical one.

5 Where Does Bias Come From?

Understanding algorithmic bias requires tracing its sources. Bias can enter at multiple stages:

5.1 Biased Training Data

Machine learning algorithms learn from historical data. If that history reflects discrimination, the algorithm learns to discriminate. Amazon's résumé screener learned from a decade of biased hiring. Facial recognition systems trained primarily on light-skinned faces perform worse on darker-skinned faces—not because of any explicit rule, but because the training data was unrepresentative.

5.2 Biased Problem Framing

Sometimes the bias is in how we define the problem. Consider an algorithm to predict "employee success." If success is measured by promotions, and promotions have historically been biased, the algorithm optimizes for a biased outcome. The algorithm might work perfectly at its stated goal while perpetuating injustice.

5.3 Proxy Variables

Even if we remove protected characteristics like race or gender from the algorithm's inputs, other variables can serve as **proxies**. ZIP codes correlate with race due to historical segregation. Names can indicate gender or ethnicity. An algorithm that has never "seen" race might still effectively use race through these proxies.

The Proxy Problem

```
// This algorithm doesn't use race directly...
public int calculateRiskScore(Applicant app) {
    int score = 50;

    // But ZIP code correlates with race
    if (isHighIncomeZipCode(app.zipCode)) score += 15;

    // Name patterns correlate with ethnicity
    if (hasCommonDefaultPattern(app.name)) score -= 10;

    // Education correlates with family wealth
    if (app.collegeRanking < 50) score += 20;

    return score;
```

```
}
// ...but it might still produce racially disparate outcomes
```

5.4 Feedback Loops

Algorithms can create self-fulfilling prophecies. If a "predictive policing" algorithm sends more officers to certain neighborhoods, more arrests occur there, which generates data suggesting those neighborhoods are high-crime, which sends even more officers. The algorithm's predictions become true *because* of the algorithm, not because of underlying reality.

6 Transparency and Accountability

Who is responsible when an algorithm causes harm? This question has no easy answer. The chain of responsibility might include the programmers who wrote the code, the data scientists who trained the model, the executives who decided to deploy it, and the organization that uses it. Often, no single person fully understands how a complex machine learning system makes its decisions.

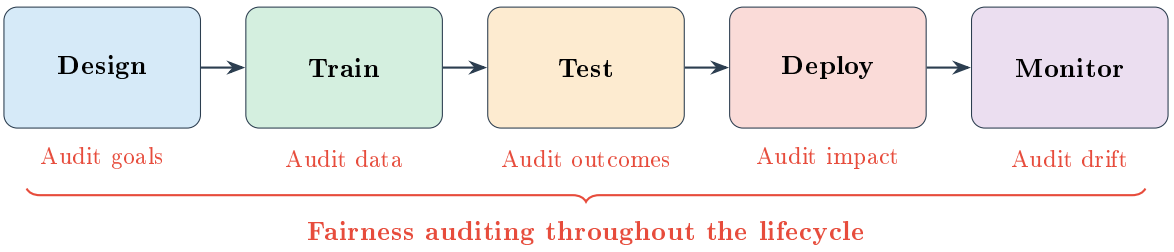
6.1 The Black Box Problem

Many modern machine learning systems are **black boxes**—their internal workings are too complex for humans to interpret. A deep neural network might have millions of parameters, and no one can explain exactly why it made a particular decision. This creates a tension between accuracy (complex models often perform better) and **explainability** (simpler models are easier to understand and audit).

Some jurisdictions now require **algorithmic transparency**. The European Union's General Data Protection Regulation (GDPR) includes a "right to explanation"—individuals can demand to know the logic behind automated decisions that significantly affect them. But what counts as an adequate explanation remains contested.

6.2 Algorithmic Auditing

One response to these challenges is **algorithmic auditing**: systematically testing algorithms for bias before and after deployment. This might involve checking whether outcomes differ across demographic groups, testing with synthetic data designed to reveal problems, or comparing algorithmic decisions to human expert judgments.



7 What Should We Do?

There are no easy answers to algorithmic fairness, but there are better and worse approaches. Here are principles that researchers and ethicists have proposed:

Context matters. The right definition of fairness depends on the domain. Equal accuracy might be paramount in criminal justice, where errors have severe consequences for individuals. Demographic parity might matter more in hiring, where we want to counteract historical exclusion.

Humans should remain in the loop. Algorithms can inform decisions, but humans should retain meaningful oversight, especially for high-stakes choices. A judge should consider a risk score as one input among many, not as a definitive answer.

Affected communities should have voice. People impacted by algorithmic systems should participate in decisions about how those systems are designed and deployed. Technical experts alone shouldn't decide what counts as "fair."

We should measure and monitor. You can't fix what you don't measure. Organizations deploying algorithms should continuously track outcomes across different groups and be prepared to adjust when problems emerge.

Trade-offs should be explicit. When perfect fairness is impossible, the choice of which fairness to prioritize should be made deliberately and transparently, not hidden in technical implementation details.

Ultimately, algorithms are tools created by humans, encoding human choices. They can perpetuate injustice or help remedy it—depending on how thoughtfully they're designed, deployed, and governed. As future programmers, you won't just write code; you'll make decisions that affect people's lives. Understanding these ethical dimensions is part of your professional responsibility.

Discussion Questions

Discussion Questions

1. **The Human Baseline:** Algorithms are often compared to an ideal of perfect fairness. But is the realistic alternative algorithmic decision-making or *human* decision-making? Humans have biases too—studies show that identical résumés receive different responses depending on whether the name sounds white or Black, male or female. Does this change how we should evaluate algorithmic bias?

2. **Choosing Fairness:** In the COMPAS case, ProPublica emphasized equal false positive rates (Black defendants were more often wrongly labeled high-risk), while Northpointe emphasized calibration (defendants with the same score had similar outcomes regardless of race). If you were a judge using such a system, which definition of fairness would matter more to you? Why?

3. **The Proxy Problem:** If removing race from an algorithm's inputs doesn't prevent racially disparate outcomes (because of proxy variables like ZIP code), what should we do? Should we remove proxies too, even if they're genuinely predictive? Should we explicitly use race to counteract the proxies? What are the trade-offs?

4. **Transparency vs. Gaming:** Making algorithms transparent allows auditing for fairness. But it also allows people to "game" the system—manipulating their inputs to get desired outcomes. How should we balance these concerns? Are there domains where transparency should be prioritized? Where it shouldn't?

5. **Responsibility:** When an algorithm causes harm—say, wrongly denying someone a loan or keeping someone in jail who shouldn't have been—who is responsible? The program-

mers? The company? The organization that deployed it? The user who relied on it? How should we structure accountability for algorithmic systems?

Key Terms

Glossary of Key Terms

Algorithm A precise sequence of steps for solving a problem, which can be executed automatically by a computer.

Algorithmic Auditing Systematic testing of algorithms to detect bias, errors, or unintended consequences.

Algorithmic Fairness The study of how to design algorithms that treat people equitably, encompassing multiple competing definitions.

Algorithmic Transparency Making the logic, data, and decision processes of algorithms open to inspection and understanding.

Bias Systematic errors or unfair preferences in algorithmic outputs, often reflecting historical discrimination in training data.

Black Box A system whose internal workings are hidden or too complex to interpret, making decisions difficult to explain.

Demographic Parity A fairness criterion requiring equal rates of positive outcomes across different demographic groups.

Equal Accuracy A fairness criterion requiring equal error rates (false positives, false negatives) across groups.

Explainability The degree to which a human can understand why an algorithm made a particular decision.

Individual Fairness A fairness criterion requiring that similar individuals receive similar treatment.

Machine Learning Algorithms that learn patterns from data rather than following explicitly programmed rules.

Proxy Variable A variable that correlates with a protected characteristic (like race or gender) and can indirectly introduce bias.

Training Data The historical examples used to teach a machine learning algorithm, which shape what patterns it learns.