What is Machine Learning?
And why might it be unfair?

Aaron Roth
Computer and Information Sciences
University of Pennsylvania
PREDICTIVE POLICING: USING MACHINE LEARNING TO DETECT PATTERNS OF CRIME
Machine Bias
There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica
May 23, 2016

ON A SPRING AFTERNOON IN 2014, Brisha Borden was running late to pick up her god-sister from school when she spotted an unlocked kid's blue Huffy bicycle and a silver Razor scooter. Borden and a friend grabbed the bike and scooter and tried to ride them down the street in the Fort Lauderdale suburb of Coral Springs.

Just as the 18-year-old girls were realizing they were too big for the tiny conveyances — which belonged to a 6-year-old boy — a woman came running after them saying, “That's my kid's stuff.” Borden and her friend immediately dropped the bike and scooter and walked away.

But it was too late — a neighbor who witnessed the heist had already called the police. Borden and her friend were arrested and charged with burglary and petty theft for the
What is Machine Learning?

It's just statistics*. 

*With a particular emphasis on prediction.
*And with an eye towards engineering in its design.
This Talk:
Focus on Supervised Classification

• And ignore 2 other major subfields of ML:
  • Unsupervised learning (Clustering)
  • Reinforcement learning (Control)
The Basic Setup

- Given: Data, consisting of \(d\) features and a label.
- Goal: Find a rule to predict label from features.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Yes</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Yes</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>No</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>No</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Yes</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>No</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\[(\text{College and Employed and not Bankruptcy}) \text{ or (Tall and Employed and not College})\]
The Basic Setup

• Given data, select a hypothesis $h: \{Y, N\}^d \rightarrow \{Y, N\}$

• Goal is not prediction on the training data, but prediction on new examples.
The Basic Setup

The example is misclassified if $h(x) \neq y$. 
The Basic Setup

• Goal: Find a classifier to minimize

\[
\text{err}(h) = \Pr_{(x,y) \sim P} [h(x) \neq y]
\]

We don’t know \(P\)...

But we *can* minimize the empirical error:

\[
\hat{\text{err}}(h, D) = \frac{1}{n} |\{ (x, y) \in D : h(x) \neq y \}|
\]
The Basic Setup

- Empirical Error Minimization:
  - Try a lookup table!
  \[ h(x) = Y \text{ if } x \in \{YNYYN, YNYYN, YNYYN, NYYYY\} \]
  \[ h(x) = N \text{ otherwise.} \]
  \[ \hat{err}(h, D) = 0. \]

- This would over-fit. We haven’t *learned*, just memorized. Learning must summarize.
The Basic Setup

• Instead, limit hypotheses to come from a “simple” class $\mathcal{C}$.
  • E.g. linear functions, or small decision trees, etc.

Compute $h^* = \arg\min_{h \in \mathcal{H}} \hat{\text{err}}(h, D)$

$$err(h^*) \leq \hat{\text{err}}(h^*, D) + \max_{h \in \mathcal{C}} |\text{err}(h) - \hat{\text{err}}(h, D)|$$

Training Error

Generalization Error
The Basic Setup

• If you have sufficiently much data to guarantee:
  \[ \text{generalization error} \leq \epsilon \]
  Then you know \( \text{err}(h^*) \leq \text{OPT}(C) + \epsilon \).

• How much is enough?
  • Depends on the complexity of the functions in \( C \).
The Basic Setup

- VC-Dimension: “The largest number of points that functions in your class can label in all possible ways”

- VC-Dimension of 2 dimensional linear functions: 3
- VC-Dimension of d-dimensional linear functions: d+1
- “Rule of Thumb”: VC-Dimension $\approx$ “number of parameters”
The Basic Setup

• If $n \geq \frac{VCDIM(C)}{\epsilon^2}$, then with high probability:
  
  Generalization error $\leq \epsilon$
  
  So, $\text{err} (h^*) \leq \text{OPT}(C) + \epsilon$.

• Have to trade off complexity of $C$ with generalization error...
  
  • Increasing complexity decreases $\text{OPT}(C)$, increases $\epsilon$.
  
  • If you want both, need more data.
The Basic Setup

• Machine Learning as Optimization:

Minimize $\sum_{i=1}^{n} \ell(h; x_i, y_i)$

such that $h \in C$

• e.g. $\ell(h; x_i, y_i) = \begin{cases} 1, & h(x_i) \neq y_i \\ 0, & h(x_i) = y_i \end{cases}$
Caveat! Computational Hardness!

• Optimizing classification accuracy often intractable.

• Solutions:
  • Optimize a different *surrogate* loss function $\hat{\ell}(h; x, y)$
    • Hinge loss, squared error, etc.
  • Heuristically try and optimize
    • Might get stuck in local optima/fail to optimize globally
  • Both

• Can still talk sensibly about generalization.

• Still looks “fair” – can observe and debate objective function, e.g.
Why might machine learning be “unfair”? 

- Many reasons: 
  - Data might encode existing biases. 
    - E.g. labels are not “Committed a crime?” but “Was arrested.” 
  - Data collection feedback loops. 
    - E.g. only observe “Paid back loan?” if the loan was granted. 
  - Different populations with different properties. 
    - E.g. “SAT score” might correlate with label differently in populations that employ SAT tutors. 
  - Less data (by definition) about minority populations.
Why might machine learning be “unfair”? 

- Many reasons:
  - Data might encode existing biases.
    - E.g. labels are not “Committed a crime?” but “Was arrested.”
  - Data collection feedback loops.
    - E.g. only observe “Paid back loan?” if the loan was granted.
  - **Different populations with different properties.**
    - E.g. “SAT score” might correlate with label differently in populations that employ SAT tutors.
  - Less data (by definition) about minority populations.
Upshot: To be “fair”, the algorithm may need to *explicitly* take into account group membership. Else, optimizing accuracy fits the majority population.
Why might machine learning be “unfair”? 

• Many reasons:  
  • Data might encode existing biases.  
    • E.g. labels are not “Committed a crime?” but “Was arrested.”  
  • Data collection feedback loops.  
    • E.g. only observe “Paid back loan?” if the loan was granted.  
  • Different populations with different properties.  
    • E.g. “SAT score” might correlate with label differently in populations that employ SAT tutors.  
• Less data (by definition) about minority populations.
Upshot: Algorithms trained on minority populations will be less accurate. Qualified individuals will be denied at a higher rate.
Why might machine learning be “unfair”? 

• Many reasons:
  • Data might encode existing biases.
    • E.g. labels are not “Committed a crime?” but “Was arrested.”
  • Data collection feedback loops.
    • E.g. only observe “Paid back loan?” if the loan was granted.
  • Different populations with different properties.
    • E.g. “SAT score” might correlate with label differently in populations that employ SAT tutors.
  • Less data (by definition) about minority populations.
Toy Model

• Two kinds of loan applicants.
  • Type 1: Pays back loan with unknown probability $p_1$
  • Type 2: Pays back loan with unknown probability $p_2$
• Initially, bank believes $p_1, p_2$ uniform in $[0,1]$
• Every day, bank makes a loan to type most likely to pay it back according to posterior distribution on $p_1, p_2$
• Bank observes if it is repaid, but not counterfactuals. Bank then updates posteriors.
Toy Model

• Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)

• Expected payoff of next loan is $\frac{s_i+1}{s_i+d_i+2}$

\[ p_1 = p_2 = \frac{1}{2} \]

\[
\begin{align*}
n_i &= 0 & n_i &= 0 \\
s_i &= 0 & s_i &= 0 \\
d_i &= 0 & d_i &= 0
\end{align*}
\]
Toy Model

• Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)

• Expected payoff of next loan is $\frac{s_i+1}{s_i+d_i+2}$

$$p_1 = p_2 = \frac{1}{2}$$

<table>
<thead>
<tr>
<th></th>
<th>0.33</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_i$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$s_i$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$d_i$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Toy Model

- Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)
- Expected payoff of next loan is $\frac{s_i + 1}{s_i + d_i + 2}$

$$p_1 = p_2 = \frac{1}{2}$$

<table>
<thead>
<tr>
<th>$n_i$</th>
<th>$s_i$</th>
<th>$d_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Toy Model

• Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)

• Expected payoff of next loan is $\frac{s_i + 1}{s_i + d_i + 2}$

\[
\begin{align*}
0.33 & \quad \begin{cases} 
n_i = 1 \\
s_i = 0 \\
d_i = 1 
\end{cases} \\
0.75 & \quad \begin{cases} 
n_i = 2 \\
s_i = 2 \\
d_i = 0 
\end{cases}
\end{align*}
\]

$p_1 = p_2 = \frac{1}{2}$
Toy Model

• Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)

• Expected payoff of next loan is $\frac{s_i + 1}{s_i + d_i + 2}$

$$p_1 = p_2 = \frac{1}{2}$$

<table>
<thead>
<tr>
<th>$n_i$</th>
<th>$s_i$</th>
<th>$d_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

$0.33$ $0.60$
Toy Model

• Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)

• Expected payoff of next loan is $\frac{s_i + 1}{s_i + d_i + 2}$

\[
p_1 = p_2 = \frac{1}{2}
\]

<table>
<thead>
<tr>
<th>$n_i$</th>
<th>$s_i$</th>
<th>$d_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0.33</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>
Toy Model

• Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)

• Expected payoff of next loan is $\frac{s_i+1}{s_i+d_i+2}$

\[ p_1 = p_2 = \frac{1}{2} \]

<table>
<thead>
<tr>
<th>$n_i$</th>
<th>$s_i$</th>
<th>$d_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

0.33 0.57
Toy Model

• Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)

• Expected payoff of next loan is $\frac{s_i + 1}{s_i + d_i + 2}$

\[ p_1 = p_2 = \frac{1}{2} \]

\[
\begin{align*}
\text{0.33} & \quad n_i = 1 & n_i = 6 \\
\text{0.33} & \quad s_i = 0 & s_i = 4 \\
& \quad d_i = 1 & d_i = 2
\end{align*}
\]
Toy Model

- Suppose bank has made $n_i$ loans to population $i$, $s_i$ have paid them back, $d_i$ have defaulted. ($n_i = s_i + d_i$)
- Expected payoff of next loan is $\frac{s_i + 1}{s_i + d_i + 2}$

\[
p_1 = p_2 = \frac{1}{2}
\]

<table>
<thead>
<tr>
<th>$n_i$</th>
<th>$s_i$</th>
<th>$d_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>0.33</td>
<td>0.55</td>
<td></td>
</tr>
</tbody>
</table>
Toy Model

Upshot: Algorithms making myopically *optimal* decisions may forever discriminate against a qualified population because of unlucky prefix.

Less myopic algorithms balance *exploration* and *exploitation* – but this has its own unfairness issues.
A Venn Diagram for An Accuracy-Fairness Tradeoff

“Racist” Association = A + B
“Biased” Association = A
“Tainted” Association = C + B
“Fair” Association = C