THE SECRETS OF MACHINE LEARNING: TEN THINGS YOU WISH YOU HAD KNOWN EARLIER TO BE MORE EFFECTIVE AT DATA ANALYSIS

Cynthia Rudin and David Carlson, October 21, 2019
MACHINE LEARNING IS EVERYWHERE

Currently, a cloud of myth and hype surrounds machine learning
Used in the right ways:
• Can make an endless number of processes more efficient
• Government more effective
• Businesses more profitable
• Medical patients more safe
Used in the wrong way, can cause serious harm!

Today’s goal is to provide a tutorial on how to see past the hype and understand what makes machine learning successful
THIS LECTURE HAS TEN TOPICS

Intro to ML

9 other topics
A CORE PROBLEM OF MACHINE LEARNING: SUPERVISED BINARY CLASSIFICATION

Predict the answer to yes/no questions:

• Will this medical patient suffer a stroke next year?
• Will this person click on this online advertisement?
• Will this criminal be arrested for another crime next year?
• Did this person say ‘yes’ to the voice assistant’s request?
APPLICATIONS OF BINARY CLASSIFICATION

- Handwriting recognition (mail, bank checks)
- Voice assistants
- Product recommendation, websearch
- Automated tagging of images
- Determine creditworthiness
- Predict whether someone will be arrested
- Spell corrections
- Identify drug targets, predict material properties
A CORE PROBLEM OF MACHINE LEARNING:
(SUPERVISED REGRESSION)

Predict real-valued outcomes:
- House prices
- 10-day rainfall forecast
- Predict pollution levels during wildfires
- Echo cancellation over the phone
- Image superresolution
#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

1) Logical
#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

1) Logical

If age=19-20 and sex=male, then predict arrest
else if age=21-22 and priors=2-3 then predict arrest
else if priors >3 then predict arrest
else predict no arrest
1) Logical
#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

1) Logical
2) Linear/Additive

2HELP2S2B Score

<table>
<thead>
<tr>
<th></th>
<th>1 point</th>
<th>1 point</th>
<th>1 point</th>
<th>1 point</th>
<th>1 point</th>
<th>1 point</th>
<th>2 points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Any cEEG pattern with Frequency <strong>2 Hz</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Epileptiform Discharges</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Patterns include [LPD, LRDA, BIPD]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Patterns Superimposed with Fast or Sharp Activity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Prior Seizure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Brief Rhythmic Discharges</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Score**

<table>
<thead>
<tr>
<th>Score</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6+</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Risk</strong></td>
<td>&lt;5%</td>
<td>11.9%</td>
<td>26.9%</td>
<td>50.0%</td>
<td>73.1%</td>
<td>88.1%</td>
<td>95.3%</td>
</tr>
</tbody>
</table>
#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

1) Logical
2) Linear/Additive
#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

1) Logical
2) Linear/Additive
3) Case-Based
#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

1) Logical
2) Linear/Additive
3) Case-Based
#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

1) Logical
2) Linear/Additive
3) Case-Based
4) Iterative Summarization
On data...
NEAREST NEIGHBORS

1-Nearest Neighbor

5-Nearest Neighbors
def tree:
    if x ≤ 0.70:
        if x ≤ 0.30:
            return -0.15
        else:
            return .33
    else:
        if x ≤ 0.96:
            return -0.27
        else:
            return -0.62
NEURAL NETWORKS

Tanh MLP

ReLU MLP
SUPPORT VECTOR MACHINES

Support Vector Regression

Outcome vs. Feature Value
OVERFITTING

Statistical learning theory explains how not to overfit in a particularly elegant way.
\begin{align*}
\{(x_i, y_i)\}_{i=1}^n & \text{ independent and identically distributed from a distribution on } \mathcal{X} \times \mathcal{Y} \\
& \\
x_i \in \mathcal{X} \subset \mathbb{R}^p \text{ are covariates} & \quad y_i \text{ are labels} \\
& \\
& \\
f \in \mathcal{F}, \ f : \mathcal{X} \to \mathcal{Y} & \\
& \\
& \\
\text{loss}(f(x), y) = \begin{cases} 
1 & \text{if } y \neq \text{sign}(f(x)) \\
0 & \text{otherwise}
\end{cases} \\
& \\
& (0-1 \text{ misclassification loss})
\end{align*}
\( \{(x_i, y_i)\}_{i=1}^n \) independent and identically distributed from a distribution on \( \mathcal{X} \times \mathcal{Y} \)

\( x_i \in \mathcal{X} \subset \mathbb{R}^p \) are covariates \hspace{1cm} \( y_i \) are labels

\( y_i \in \mathbb{R} \) for regression

\( f \in \mathcal{F}, \; f : \mathcal{X} \rightarrow \mathcal{Y} \)

\[
\text{loss}(f(x), y) = (y - f(x))^2 \quad \text{(Squared loss for regression)}
\]
TrueRisk\( (f) := \mathbb{E}_{x,y} \text{loss}(f(x), y) \)  \hspace{1cm} \text{Test Error (can’t calculate)}

EmpRisk\( (f) := \frac{1}{n} \sum_{i=1}^{n} \text{loss}(f(x_i), y_i) \) \hspace{1cm} \text{Training Error (can calculate)}
The diagram illustrates the relationship between model complexity and error rates, specifically focusing on the trade-offs between training error and test error.

- **Error** is plotted on the vertical axis, with lower values indicating better performance.
- **Underfitting** occurs when the model is too simple to capture the underlying patterns in the data, leading to high training and test errors.
- **Good Models** represent a balance where both training and test errors are low.
- **Overfitting** occurs when the model is too complex, fitting the training data too well but failing to generalize to new data, resulting in low training error and high test error.

The diagram highlights the distinction between **Test Error** (True Risk) and **Training Error** (Empirical Risk), showing how the complexity of models should be tuned to achieve optimal performance.
\[ f \in \mathcal{F}, \ f : \mathcal{X} \rightarrow \mathcal{Y} \]

Assume (for now) that \( \mathcal{F} \) is finite, \( |\mathcal{F}| \) is well-defined.
$|\mathcal{F}|$ is small

$|\mathcal{F}|$ is large
Theorem (Occam's Razor Bound). With probability at least $1-\delta$ over the random draw of the training data, for all functions $f$ in function class $\mathcal{F}$ of functions mapping $\mathcal{X}$ to $\mathcal{Y}$, the following holds:

$$\text{TrueRisk}(f) \leq \text{EmpRisk}(f) + \sqrt{\frac{\log(|\mathcal{F}|) + \log(1/\delta)}{2n}}$$

Can't calculate this

Can calculate all of this
$|\mathcal{F}|$ is small

$|\mathcal{F}|$ is large
Theorem (Occam's Razor Bound). With probability at least $1-\delta$ over the random draw of the training data, for all functions $f$ in function class $\mathcal{F}$ of functions mapping $\mathcal{X}$ to $\mathcal{Y}$, the following holds:

$$\text{TrueRisk}(f) \leq \text{EmpRisk}(f) + \sqrt{\frac{\log(|\mathcal{F}|) + \log(1/\delta)}{2n}}$$

Can’t calculate this

Can calculate all of this
Occham’s Razor in a nutshell:

Training accuracy + Simplicity $\rightarrow$ Good Test Performance
REGULARIZED RISK MINIMIZATION

\[
\min_{f \in \mathcal{F}} \left( \frac{1}{n} \sum_{i=1}^{n} \text{loss}(f(x_i), y_i) + \text{complexity}(f) \right)
\]

Each ML algorithm navigates this in different ways.
BACK TO ALGORITHMS
For decision trees

\[
\min_{f \in \mathcal{F}} \left( \frac{1}{n} \sum_{i=1}^{n} \text{loss}(f(x_i), y_i) + \text{complexity}(f) \right)
\]

\[
\min_{\text{trees} \in \text{Trees}} \left( \frac{1}{n} \sum_{i=1}^{n} 1_{y_i \neq \text{tree}(x_i)} + \text{number of leaves of (tree)} \right)
\]
\[
\text{loss}(f(x), y) = \begin{cases} 
1 & \text{if } y \neq \text{sign}(f(x)) \\
0 & \text{otherwise} 
\end{cases}
\]

\[
= 1 \quad \text{if } yf(x) < 0
\]

The margin is \(yf(x)\)
The margin of \((x_i, y_i)\) with respect to \(f\) is \(y_i f(x_i)\).
Misclassification error

Misclassified

Correctly classified

Large negative margin

Small negative margin

Small positive margin

Large positive margin

$yf(x)$
Large positive margin
Small positive margin
Small negative margin
Large negative margin

Misclassification error
Exponential loss
Logistic loss
SVM Hinge loss

$yf(x)$
Misclassification error

**Exponential loss**

\[ 1_{y_i \neq \text{sign}(f(x_i))} \leq e^{-y_if(x_i)} \]

**Logistic loss**

\[ 1_{y_i \neq \text{sign}(f(x_i))} \leq \log(1 + e^{-y_if(x_i)}) \]

**SVM Hinge loss**

\[ 1_{y_i \neq \text{sign}(f(x_i))} \leq \max(0, 1 - y_if(x_i)) \]
Boosted Decision Trees minimize the exponential loss, where $f$ is a weighted sum of decision trees.

$$\mathbf{1}_{y_i \neq \text{sign}(f(x_i))} \leq e^{-y_i f(x_i)}$$

**Exponential loss**
Boosted Decision Trees minimize the exponential loss, where $f$ is a weighted sum of decision trees.
Boosted Decision Trees minimize the exponential loss, where $f$ is a weighted sum of decision trees.

Random Forests take majority vote over many different decision trees.
LINEAR / ADDITIVE MODELS
LINEAR / ADDITIVE MODELS

\[ f(x_i) = \sum_j b_j x_{ij} \quad \text{Linear models} \]

\[ f(x) = \sum_j b_j g_j(x_{.j}) \quad \text{Additive models} \]
LINEAR / ADDITIVE MODELS

\[ f(x_i) = \sum_j b_j x_{ij} \]  \hspace{0.5cm} \text{Linear models}

\[ f(x) = 5 \text{ age} + 2 \text{ prior strokes} + 3 \ldots \]

\[ f(x) = \sum_j b_j g_j(x_{.j}) \]  \hspace{0.5cm} \text{Additive models}

\[ f(x) = 5 \text{ function(age)} + 2 \text{ function (prior strokes)} + 3 \ldots \]
Logistic regression minimizes the logistic loss, where $f$ is a weighted sum of features.

$$\min_{\{b_j\}_j} \log(1 + e^{-y_i f(x_i)})$$

Logistic loss

where

$$f(x_i) = \sum_j b_j x_{ij}$$
Classify "0" vs "1"

\[ f(x_i) = \sum_j b_j x_{ij} \]
LINEAR / ADDITIVE MODELS

\[ f(x_i) = \sum_j b_j x_{ij} \]  \quad \text{Linear models}

\[ f(x) = 5 \text{ age} + 2 \text{ prior strokes} + 3 \ldots \]

\[ f(x) = \sum_j b_j g_j(x_j) \]  \quad \text{Additive models}

\[ f(x) = 5 \text{ function(age)} + 2 \text{ function (prior strokes)} + 3 \ldots \]
## LINEAR / ADDITIVE MODELS

**ATRIA stroke risk score**

<table>
<thead>
<tr>
<th>Factor</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>History of stroke</td>
<td>8</td>
</tr>
<tr>
<td>Age &lt; 65 years</td>
<td>0</td>
</tr>
<tr>
<td>Age 65-74 years</td>
<td>3</td>
</tr>
<tr>
<td>Age 74-84 years</td>
<td>5</td>
</tr>
<tr>
<td>Age ≥ 85 years</td>
<td>6</td>
</tr>
<tr>
<td>Female</td>
<td>1</td>
</tr>
<tr>
<td>Diabetes Mellitus</td>
<td>1</td>
</tr>
<tr>
<td>Past Congestive Heart Failure</td>
<td>1</td>
</tr>
<tr>
<td>History of Hypertension</td>
<td>1</td>
</tr>
<tr>
<td>Proteinuria</td>
<td>1</td>
</tr>
<tr>
<td>eGFR &lt;45 or renal disease</td>
<td>1</td>
</tr>
</tbody>
</table>

**Total score**

(0-5 points is Low risk)  
(6 points is Intermediate risk)  
(7+ points is High risk)

*Singer et al. 2013*
Misclassification error

\[ 1_{y_i \neq \text{sign}(f(x_i))} \leq e^{-y_i f(x_i)} \quad \text{Exponential loss} \]

\[ 1_{y_i \neq \text{sign}(f(x_i))} \leq \log(1 + e^{-y_i f(x_i)}) \quad \text{Logistic loss} \]

\[ 1_{y_i \neq \text{sign}(f(x_i))} \leq \max(0, 1 - y_i f(x_i)) \quad \text{SVM Hinge loss} \]
\[ 1_{y_i \neq \text{sign}(f(x_i))} \leq \max(0, 1 - y_i f(x_i)) \] SVM Hinge loss
SUPPORT VECTOR MACHINES

SVM Hinge loss

\[ \min_{\{b_j\}_j} \max(0, 1 - y_i f(x_i)) + C \sum_j b_j^2 \]

where

\[ f(x) = \sum_i b_i e^{-\frac{||x - x_i||^2}{\sigma^2}} \]

These are radial basis function kernels.
SVM with large RBF kernel
SVM with not-as-large RBF kernel
SVM with small RBF kernel
SVM with way-too-small RBF kernel
NEURAL NETWORKS

Logistic loss

$$\min_{f \in \mathcal{F}} \log(1 + e^{-y_i f(x_i)}) + \text{function of network weights}$$

Neural networks
**NEURAL NETWORKS ARE ITERATIVE SUMMARIZATION**

Neural networks use a sequence of “layers” that combine to make a prediction function:

\[
f(x_i) = \sum_j c_j h_{M}^{wM} \left( ... h_{2}^{w2} \left( h_{1}^{w1}(x_i) \right) \right)
\]

All weights are learned from data.
CONVOLUTIONAL NEURAL NETWORKS
REUSE FILTERS

Reuse same filter weights

Filter

Image

Output Layer

Fill in the output layer by shifting the filter
EACH OUTPUT MAP COMES FROM A DIFFERENT FILTER

Many features maps in the output layers
A CNN is structured in layers like the multi-layer perceptron.

CNNs follow the same iterative summarization

\[ f(x_i) = \sum_j c_j h_M^{w_M} \left( ... h_i^{w_i} \left( h_1^{w_1}(x_i) \right) \right)_j \]

Only the form of the individual functions varies.
LEARNING THE PARAMETERS

Almost all neural networks are use stochastic gradient methods, i.e.

$$\theta^{k+1} \leftarrow \theta^k - \alpha \nabla_{\theta} \ell(y, f_{\theta^k}(x_j)),$$

where $j$ is a random data entry.

Adding momentum (add in decaying historical gradients) is very popular.
NEURAL NETWORKS

Tanh MLP

<table>
<thead>
<tr>
<th>Feature Value</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>-0.8</td>
</tr>
<tr>
<td>0.2</td>
<td>-0.6</td>
</tr>
<tr>
<td>0.4</td>
<td>-0.2</td>
</tr>
<tr>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>1.0</td>
<td>0.6</td>
</tr>
</tbody>
</table>

ReLU MLP

<table>
<thead>
<tr>
<th>Feature Value</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>-0.8</td>
</tr>
<tr>
<td>0.2</td>
<td>-0.6</td>
</tr>
<tr>
<td>0.4</td>
<td>-0.2</td>
</tr>
<tr>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4</td>
</tr>
<tr>
<td>1.0</td>
<td>0.6</td>
</tr>
</tbody>
</table>
PARAMETER TUNING

- Makes a huge difference!
#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

- Regularized risk minimization, the principle of Occam’s razor: 
  Training accuracy + Simplicity $\rightarrow$ Good Test Performance

- Algorithms and forms of models:
  - **Logical**: decision trees
  - **Linear combinations**: ensembles (boosted decision trees, random forest), logistic regression / linear models, additive models, support vector machines
  - **Iterative Summarization**: neural networks
  - **Case-Based Reasoning and Kernel Methods**: k-nearest neighbors, support vector machines
SO FAR: #1 (OUT OF “TOP 10”)

#1: ML ALGORITHMS CAN BE SPLIT INTO 4 (OR MORE) MAIN FAMILIES

- Intro to ML
- 9 other topics
#2: ALL MACHINE LEARNING METHODS PERFORM SIMILARLY (WITH SOME CAVEATS)
#2: ALL MACHINE LEARNING METHODS PERFORM SIMILARLY (WITH SOME CAVEATS)

- CNN’s are excellent for computer vision, but provide no advantage on a huge number of data science problems.

- Most ML methods perform similarly (if tuned) when features have inherent meaning (e.g., age, gender, blood pressure), as opposed to raw pixel values.
COMPUTER VISION PROBLEMS HAVE SPECIAL PROPERTIES

- Natural images live on a narrow manifold in pixel space, whereas most structured data do not.
- Neighboring pixels in natural images tend to be closely related to each other.
- Leveraging this information is critical to good performance (CNNs and deep learning methods).
IN OTHER PROBLEMS...

When neural networks do not have an advantage,
- adding more data,
- adding domain knowledge,
- or improving the quality of data
are usually more valuable than trying a different algorithm or changing tuning procedures.

Adding in domain knowledge in various ways can be more powerful than anything else!
SOME RECOMMENDATIONS

If you have classification or regression data with inherently meaningful (non-raw) covariates, try several algorithms. If some of them all perform similarly after parameter tuning, use the simplest or most meaningful model.

Try to embed domain knowledge into the model.

Don’t use neural networks simply to state that you are “doing deep learning.”
#3: NEURAL NETWORKS ARE HARD TO TRAIN
#3: NEURAL NETWORKS ARE HARD TO TRAIN

Many choices in neural networks:

- How many layers?
- How complex is each layer?
- How can we effectively learn the network?

The objective function is highly non-convex and lacks smooth gradients…
NEURAL NETWORK STRUCTURE IS COMPLEX

Krizhevsky et al 2012.
BLOCK STRUCTURES ARE COMPLEX

A Inception-Resnet-v2 block structure from Szegedy et al. 2017

Structure of the Long-Short Term Memory Block (LSTM)
LEARNED WITH STOCHASTIC GRADIENT METHODS + A LOT OF “TRICKS”

Momentum terms in the gradient updates
Adaptive preconditioning
Smart initializations
Early stopping
Data augmentation
Transfer learning
EARLY STOPPING

Training Loss Keeps Improving

Best Validation (and Generalization) Here

Average Loss vs. Iterations

Training Loss Keeps Improving

Training
Validation
DATA AUGMENTATION

Zooms

Shifts

Flips
TRANSFER LEARNING

Leverage information from one problem to help solve another.
TRANSFER LEARNING

Feature detectors at the top of the network are typically highly specialized for a particular task.

Healthy or Diseased?

Layer 3 Filters

Layer 2 Filters

Layer 1 Filters

Input Image

Layer 1
Feature “Maps”

Layer 2
Feature “Maps”

Layer 3
Feature “Maps”

Layer 1

Layer 2

Layer 3
TRANSFER LEARNING

Early feature weights
- highly similar between applications
- can be initialized with “pre-trained” weights

Healthy or Diseased?

Layer 3 Filters
Layer 3 Feature “Maps”
Layer 2 Filters
Layer 2 Feature “Maps”
Layer 1 Filters
Layer 1 Feature “Maps”
Input Image
#4: WITHOUT INTERPRETABILTY, YOU CAN MAKE BAD MISTAKES
#4: WITHOUT INTERPRETABILITY, YOU CAN MAKE BAD MISTAKES

“Black box models” refers to models that are too complicated to understand, or proprietary models. These models were built for low stakes decisions, such as handwriting recognition, web image classification, and online advertising.

In high stakes decisions, lack of trust is a major issue for machine learning methods, and it should be, as training data are often flawed in unknown ways.
ISSUES DUE TO DATA QUALITY

There have been some high profile cases where:

- incorrect data entered into a proprietary model has led to denial of parole
- Incorrect data entered into a model has led to bad bail decisions
- Medical predictions were being made based on the equipment used to make the measurements.
SHOULD TRY TO BUILD AN INTERPRETABLE MODEL

Most people using machine learning do not make an attempt to create an interpretable model.

They believe that they would need to sacrifice accuracy to gain interpretability.

(Often we will pay no penalty for interpretability – #6)
#5: EXPLANATIONS CAN BE MISLEADING
#5: EXPLANATIONS CAN BE MISLEADING

Building an interpretable model is a world different than explaining a black box.
### SALIENCY MAP AS EXPLANATION?

<table>
<thead>
<tr>
<th>Test Image</th>
<th>Evidence for Animal Being a Golden Retriever</th>
<th>Evidence for Animal Being a Tennis Ball</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Test Image" /></td>
<td><img src="image2" alt="Evidence" /></td>
<td><img src="image3" alt="Evidence" /></td>
</tr>
</tbody>
</table>
#5: EXPLANATIONS CAN BE MISLEADING

Building an interpretable model is a world different than explaining a black box.

Because variables are often correlated, explanation models often depend on different variables than the original black box model.

Several reasons to be cautious about explanations
Explanations must be wrong. Otherwise the explanation’s predictions would be equal to the black box’s predictions and one would thus not need the black box at all. If the explanation differs from the black box often, one cannot trust the explanation, and thus, one cannot trust the black box.
DOUBLE TROUBLE

The combination of a black box model and an explanation model force the model’s developer to troubleshoot two models rather than one.

Instead, can we create a single, interpretable model?
#6: THERE OFTEN EXISTS AN ACCURATE—YET-INTERPRETABLE MODEL
#6: THERE OFTEN EXISTS AN ACCURATE-YET-INTERPRETABLE MODEL

...even for neural networks

Incorporate a constraint on the model’s interpretability:

\[
\min_f \frac{1}{n} \sum_{i=1}^{n} \text{loss}(f(x_i), y_i)
\]

s. t. \(\text{InterpretabilityPenalty}(f) < C\)

How can we define such an interpretability penalty?
ONE APPROACH: PENALTY ON NUMBER OF FEATURES USED

2HELPS2B score for predicting seizures in ICU patients (Struck et al 2017), constructed by the RiskSLIM ML algorithm (Ustun & R 2019). The factors and point scores were chosen (by an algorithm)
An interpretable decision tree to predict whether an individual will be arrested in the future. Hu et al. NeurIPS 2019
CAN WE MAKE AN INTERPRETABLE NEURAL NETWORK?
“THIS LOOKS LIKE THAT”

(Chen et al. NeurIPS 2019)
Why is this bird classified as a red-bellied woodpecker?

Evidence for this bird being a red-bellied woodpecker:

<table>
<thead>
<tr>
<th>Original image (box showing part that looks like prototype)</th>
<th>Prototype</th>
<th>Training image where prototype comes from</th>
<th>Activation map</th>
<th>Similarity score</th>
<th>Class connection</th>
<th>Points contributed</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Original image" /></td>
<td><img src="image2" alt="Prototype" /></td>
<td><img src="image3" alt="Training image" /></td>
<td><img src="image4" alt="Activation map" /></td>
<td>6.499</td>
<td>1.180</td>
<td>7.669</td>
</tr>
<tr>
<td><img src="image5" alt="Original image" /></td>
<td><img src="image6" alt="Prototype" /></td>
<td><img src="image7" alt="Training image" /></td>
<td><img src="image8" alt="Activation map" /></td>
<td>4.392</td>
<td>1.127</td>
<td>4.950</td>
</tr>
<tr>
<td><img src="image9" alt="Original image" /></td>
<td><img src="image10" alt="Prototype" /></td>
<td><img src="image11" alt="Training image" /></td>
<td><img src="image12" alt="Activation map" /></td>
<td>3.890</td>
<td>1.108</td>
<td>4.310</td>
</tr>
</tbody>
</table>

Total points to red-bellied woodpecker: 32.736

Evidence for this bird being a red-cockaded woodpecker:

<table>
<thead>
<tr>
<th>Original image (box showing part that looks like prototype)</th>
<th>Prototype</th>
<th>Training image where prototype comes from</th>
<th>Activation map</th>
<th>Similarity score</th>
<th>Class connection</th>
<th>Points contributed</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image13" alt="Original image" /></td>
<td><img src="image14" alt="Prototype" /></td>
<td><img src="image15" alt="Training image" /></td>
<td><img src="image16" alt="Activation map" /></td>
<td>2.452</td>
<td>1.046</td>
<td>2.565</td>
</tr>
<tr>
<td><img src="image17" alt="Original image" /></td>
<td><img src="image18" alt="Prototype" /></td>
<td><img src="image19" alt="Training image" /></td>
<td><img src="image20" alt="Activation map" /></td>
<td>2.125</td>
<td>1.091</td>
<td>2.318</td>
</tr>
<tr>
<td><img src="image21" alt="Original image" /></td>
<td><img src="image22" alt="Prototype" /></td>
<td><img src="image23" alt="Training image" /></td>
<td><img src="image24" alt="Activation map" /></td>
<td>1.945</td>
<td>1.069</td>
<td>2.079</td>
</tr>
</tbody>
</table>

Total points to red-cockaded woodpecker: 16.886
#7: SPECIAL PROPERTIES MUST BE BUILT IN
#7: SPECIAL PROPERTIES MUST BE BUILT IN

If we want special properties, such as:

- Interpretability
- Robustness
  - Robust to shifts in distribution
  - Spatial invariance
  - Robust to drifting sensor calibration
- Fairness
  - Defined how?

These must be *defined* and built-in to the system (rarely do these happen off-the-shelf)
WHAT IS FAIRNESS IN ML?

Our training data is our historical data; historical decisions often contain sources of bias, and a ML system will “learn” these biases. Could enforce fairness algorithmically (e.g., over gender or race):

- Requires modifying the ML method
- Requires a mathematical definition of fairness—no small feat!
CAUSAL INFERENCE IS DIFFERENT THAN PREDICTION
#8: CAUSAL INFERENCE IS DIFFERENT THAN PREDICTION

Correlation does not imply causation, so high feature importance doesn’t imply causation.

There is a rigorous statistical framework for causal inference.
What to know the effect of taking a drug ($T = 1$) versus not taking a drug ($T = 0$) in a particular patient.

Want to predict the conditional average treatment effect (CATE), $E_Y(Y|T = 1, X = x) - E_Y(Y|T = 0, X = x)$

Can we just use ML to estimate the two outcomes? Not without making critical assumptions.
CRITICAL ASSUMPTIONS

Strong Ignorability: we have not left out any important covariates, and that the probability of being treated is between 0 and 1 but not either exactly 0 or 1

SUTVA: the stable unit treatment value assumption, that the outcome of one unit does not affect the treatment of another
Caruana et al. (2015) created an interpretable model that having asthma lowers the risk of dying from pneumonia. Counterintuitive!

Issue was that there was an unobserved confounder; patients with asthma received more intensive care.
#9: NEURAL NETWORK ARCHITECTURES: THERE IS A METHOD TO THE MADNESS, BUT NOT ALWAYS

Reading deep learning papers make it seem like absolute madness in how to construct a deep network.

Networks are made up of highly reused “building blocks” and overall architectures.

Why do so many people use similar networks?
• Training neural networks is computationally challenging
• *Transfer learning* is so important that we are critically dependent on pre-trained networks
REVISITING TRANSFER LEARNING

Most problems cannot effectively retrain a full complicated network from scratch!
MODERN DEEP LEARNING REUSES LOTS OF THE SAME BLOCK STRUCTURES

Essentially all deep networks are based on a few key blocks:

- Dense
- Convolutional
- Simple Recurrent
- Long-Short Term Memory
- Skip-Connections

More complicated blocks typically just combine these initial blocks.
#10: MACHINE LEARNING MUST "LEARN" FROM DATA—IT CAN'T DO ANYTHING.
#10: IT IS A MYTH THAT ARTIFICIAL INTELLIGENCE CAN DO ANYTHING.

In many specialized cases, current machine learning methods are outperforming humans.

These systems are highly specific: models trained to recognize certain objects can only recognize these objects.

How much do our data represent the real world?
#10: IT IS A MYTH THAT ARTIFICIAL INTELLIGENCE CAN DO ANYTHING.

ML can be brittle: Adversarial attacks show that changing a single pixel in an image can change the predicted class in modern ML systems.

Small modifications to the real world can trick ML systems (e.g., adding a small sticker to a stop sign)
GENERATING DATA IS STILL LIMITED BUT QUICKLY IMPROVING

Face generation is now at photo-realistic levels.

Voice generation recently used to defraud companies into wire transferring large sums of money

Karras et al, 2019
ML SYSTEMS ARE STILL NARROW

Many ML applications are actually narrow tasks

• Every new task requires new (and typically abundant) training data
TOP TEN

#1: ML algorithms can be split into 4 (or more) main families
#2: All machine learning methods perform similarly (with some caveats)
#3: Neural Networks Are Hard to Train
#4: Without Interpretability, You Can Make Bad Mistakes
#5: Explanations Can Be Misleading
#6: There Often Exists an Accurate-Yet-Interpretable Model
#7: Special Properties Must Be Built In
#8: Causal Inference is Different than Prediction
#9: Neural Network Architectures: A Method to the Madness
#10: Machine Learning Must “Learn” from data—it can’t do anything