The mathematics of machine learning and deep learning

Sanjeev Arora
Princeton University and Institute for Advanced Study

http://www.cs.princeton.edu/~arora/
Group website: unsupervised.princeton.edu
Blog: www.offconvex.org
Twitter: @prfsanjeevarora

Support: NSF, ONR, Simons Foundation, Schmidt Foundation, Amazon Research, Mozilla Research. DARPA/SRC
Machine learning (ML): A new kind of science

“Science of creating machines/programs that improve from experience and interaction.”

(Imitating human intelligence not an explicit goal.)
What is Learning?
What does it mean to understand the world?
("What sequence of pixels correspond to a scene with a pedestrian?")
Talk overview

• Part 1: Mathematical formulation of Machine Learning (ML).

• Part 2: ML in action (unsupervised, sequential decision-making)

• Part 3: Toward mathematical understanding of deep learning

• Part 4: Taking Stock/Concluding thoughts

“A tour with many stops…”
Part 1

Mathematical formalization of Machine Learning (ML)

(Operationally speaking, boils down to learning patterns in data.)
Old Idea: Curve fitting (Legendre, Gauss, c. 1800)

Phillips curve (1958):

Inflation

Unemployment

Gas Law (c. 1800)

\[ PV = nRT \]

Machine Learning uses surface fitting, with many more variables (eg 20M) ("Learning patterns in data.")
Example: Learning to score reviews

Score = −1.5
“Slow moving. Couldn’t get into this movie despite all the awards it has won.”

Score = 2.5
“Wow! Did not know what to expect and was delighted! Loved the homage paid to the musicals of old. “

Given: Reviews for different movies and rating score (-3 to +3).
Task: Learn to predict rating score given text of a new review.

The “law” of movie review scores??
Example: Learning to rate reviews (contd)

“Slow moving. Couldn’t get into this movie despite all the awards it has won.”

100,000 words in English dictionary.

Hypothesized “law” (simple linear model):

- Each word $w$ has sentiment score $\theta_w \in \mathbb{R}$
  
  Review score $\approx$ total sentiment score of all words in it

Finding sentiment score for 100,000 words $\equiv$ Surface fitting with 100,000 variables!
100,000 words in English dictionary.

Hypothesized “law” (simple linear model):

- Each word $w$ has sentiment score $\theta_w$

Review rating $\approx$ total sentiment score of all words in it

$$\text{Min} \sum_{\text{review}} \left( \text{Score} - \sum_{w \in \text{review}} \theta_w \right)^2$$

“Loss function”

Algorithm: “Gaussian Least Square Fit”; (Solvable in seconds given few million ranked reviews.)
Hypothesized law:

- Each word $w$ has sentiment score $\theta_w$
  
  Review rating $\approx$ total sentiment score of all words in it

<table>
<thead>
<tr>
<th>Word</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>as</td>
<td>+0.1</td>
</tr>
<tr>
<td>good</td>
<td>+0.6</td>
</tr>
<tr>
<td>homage</td>
<td>0.0</td>
</tr>
<tr>
<td>loved</td>
<td>+1.1</td>
</tr>
<tr>
<td>musicals</td>
<td>+0.1</td>
</tr>
<tr>
<td>of</td>
<td>-0.1</td>
</tr>
<tr>
<td>old</td>
<td>-0.4</td>
</tr>
<tr>
<td>paid</td>
<td>-0.3</td>
</tr>
<tr>
<td>story</td>
<td>+0.3</td>
</tr>
<tr>
<td>the</td>
<td>+0.1</td>
</tr>
<tr>
<td>to</td>
<td>-0.1</td>
</tr>
<tr>
<td>was</td>
<td>-0.2</td>
</tr>
<tr>
<td>well</td>
<td>+1.4</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

“Loved the homage paid to the musicals of old. Story was good as well.”

“Discovered Patterns” = How words correlate with positive score
ML \approx \text{finding suitable function ("model") given examples of desired input/output behavior}

\[ f_\theta(X) \]

\( \theta \in \mathbb{R}^d \)

(trainable parameters)

\((X, Y) : \text{(Input, Output) pair}\)

\[ \sum_{w \in \text{review}} \theta_w \]

Score

Training method:

\[
\text{Min} \sum_{\text{review}} \left( \text{Score} - \sum_{w \in \text{review}} \theta_w \right)
\]

Desired output

Model’s output
ML \approx \text{finding suitable function ("model") given examples of desired input/output behavior}

\[ f_\theta(X) \]

\[ X \rightarrow \theta \rightarrow Y \]

\[ \theta \in \mathbb{R}^d \]

(trainable parameters)

Training data: \{(X^{(i)}, Y^{(i)}): i=1,\ldots,N\}

"Loss" \[ \ell(\theta) = \sum_{i=1}^{N} \left( f_\theta(X^{(i)}) - Y^{(i)} \right)^2 \]

(Many other loss formulations exist....)

Prev. Example:

Review text

\[ \theta \rightarrow \sum_{w \in \text{review}} \theta_w \rightarrow \text{Score} \]

Training method:

\[
\begin{align*}
\text{Min} & \quad \sum_{\text{review}} \left( \text{Score} - \sum_{w \in \text{review}} \theta_w \right) \\
\text{Desired output} & \quad \text{Model’s output}
\end{align*}
\]
Formal framework

\[ X \xrightarrow{\theta} f_{\theta}(X) \xrightarrow{\theta} Y \]

\( \theta \in \mathbb{R}^d \)  
(trainable parameters)

Training data: \( \{(X^{(i)}, Y^{(i)}): i=1,..,N\} \)

“Loss”  
\[
\ell(\theta) = \sum_{i=1}^{N} (f_{\theta}(X^{(i)}) - Y^{(i)})^2
\]

(Many other loss formulations exist....)

(inherited from classical statistics)

Assumption: Samples \( (X^{(i)}, Y^{(i)}) \) drawn from a probability distribution \( D \) (e.g., distribution on pairs “(movie review, rating score)”)

TRAINING: Optimize loss \( \ell(\theta) \).

TESTING: Take new \( X \)’s and see how well the trained model predicts the missing \( Y \) (e.g., “given review, predict rating score”)

In practice: TRAIN on 80% of data;  
TEST by evaluating loss on remaining 20%
Formal framework

\[ \theta \in \mathbb{R}^d \]

(Training data: \( \{(X^{(i)}, Y^{(i)}): i=1,..,N\} \))

"Loss" \[ \ell(\theta) = \sum_{i=1}^{N} (f_\theta(X^{(i)}) - Y^{(i)})^2 \]

(Many other loss formulations exist....)

Assumption: Samples \((X^{(i)}, Y^{(i)})\) drawn from a probability distribution \(D\) (e.g., distribution on pairs "(movie review, rating score)"

Comment: Fourier analysis allows learning \(f\) from \((x, f(x))\) examples.....

(under reasonable assumptions on \(f\)....)

But practically infeasible: If \(X \in \mathbb{R}^n\), need \(\exp(n)\) samples of \((X, f(X))\) and also \(\exp(n)\) computation time.

(In practice \(n = 10^4\) say, and \# samples = 10n)
Training via Gradient Descent ("natural algorithm")

\[ \theta \in \mathbb{R}^d \]

Trainable parameters

Starting with some \( \theta^{(0)} \),
compute for \( t=0,1,2,.. \)

\[ \theta^{(t+1)} = \theta^{(t)} - \eta \nabla (\ell) \]

(\( \eta \in \mathbb{R} \) is "learning rate", say, 0.01)

Often nonconvex, esp. in deep learning.

Loss \( \ell(\theta) \)

Deficiency in desired fit

In practice can improve GD with tricks: time-varying \( \eta \), past gradients ("momentum"), "regularization", ..
Subcase: deep learning*  (deep models = “multilayered”)

\[ \theta: \{ M_1, M_2, \ldots \} \]

```
Input
X_1

\[ M_1 \]

Matrix

\[ M_1X_1 \]

Nonlinearity

\[ X_2 \]

\[ M_2 \]

Matrix

\[ M_2X_3 \]

Nonlinearity

\[ X_3 \]

\[ M_3 \]

Matrix

\[ \ldots \]

\[ \ldots \]

\[ \text{Output} \]

\[ f_\theta(X_1) \]
```

“Nonlinearity”: Given a vector, output same vector but negative entries turned to zero.

Training data: \{(X_1^{(i)}, Y^{(i)}): i=1,..,N\}

\[ \ell(\theta) = \sum_{i=1}^{N} (f_\theta(X_1^{(i)}) - Y^{(i)})^2 \]

How to compute gradient of Loss??

Backpropagation Algorithm does it fast; clever application of chain rule [Werbos’77, Rumelhart et al’84]

(*Highly simplistic: could have “convolution”, “bias”, “skip connections”, other loss fns etc.)
Subcase: deep learning*

(deep models = “multilayered”)

\[ \theta: \{ M_1, M_2, \ldots \} \]

Input \( X_1 \)

Matrix \( M_1 \)

\( M_1 X_1 \)

\( X_2 \)

Matrix \( M_2 \)

\( M_2 X_3 \)

\( X_3 \)

Matrix \( M_3 \)

\( \cdot \cdot \cdot \)

Output \( f_\theta(X_1) \)

“Nonlinearity”: Given a vector, output same vector but negative entries turned to zero.

Training data: \( \{(X_1^{(i)}, Y^{(i)}): i = 1, \ldots, N\} \)

Nonlinear enough to express many things; linear enough to allow quick optimization on today’s computers

(*Highly simplistic: could have “convolution”, “bias”, “skip connections”, other loss fns etc.)
Part 2

Machine Learning in Action (Unsupervised, Interactive, etc.)

(Key idea: need to define “loss function” creatively....)
Review rating = total sentiment score of all words in it

\[ \sum_{w \in \text{review}} \theta_w \]

This is not how we humans do it!

(i) Word order in text matters. (“No good” vs “Good, no?”)
(ii) Don’t need millions of examples. (Because we understand language!)

Science has long tried to understand and formalize language + semantics (using logic, grammars, model theory, …)

How can machines “understand” language?
(Arguably, more general-purpose than rating reviews!)
Unsupervised learning  (no human-supplied labels)

• Key idea: Using large corpus (eg, Wikipedia), train model to predict part of text from adjacent text.

  Example: “I went to a café and ordered a…. “

Model learns to do such completions by training on huge amounts of text…. 

(In the process, implicitly picks up on grammar rules, common sense etc. )

Used in: Machine Translation, Question-Answering (e.g. Siri, Alexa..)
A Language model (baby “word2vec” [Mikolov et al’13])

Preceding words $w_1, w_2, \ldots, w_5$  

Distribution on all English words $Pr[w|w_1 \ldots w_5] \propto \exp\left(\frac{1}{5} \sum_i \langle v_w, v_{w_i} \rangle \right)$

(implicit normalizing term; will ignore for simplicity)

Measure of goodness of fit? (i.e., loss function)?
A Language model (baby "word2vec" [mikolov et al’13])

Preceding words \(w_1, w_2, \ldots, w_5\) → \(\theta\) → Distribution on all English words

\(\theta = \{v_w \in \mathbb{R}^{300} : w \text{ an English word}\}\)

("semantic vectors")

Loss \(\ell(\theta)\): Reciprocal of Probability assigned by model to Wikipedia = \(w_1 w_2 w_3 \ldots w_N\)

\[
\prod_{i=6}^{N} \Pr[w_i | w_{i-5}, \ldots, w_{i-1}]
\]

\[
= \exp\left(\sum_{i=6}^{N} \sum_{j=1}^{5} \frac{1}{5} \langle v_{w_i}, v_{w_{i-j}} \rangle\right)
\]

(* Omitting negative sampling term…)*
Properties of semantic word vectors

Cosine of angle captures human estimates of “similarity” [Deerwester et al’90]

(Possible to use word vectors to define sentence/paragraph vectors that capture sentence/paragraph similarity [“SIF embeddings” [A., Liang, Ma,’17])

Incorporating semantic vectors improves movie rating task …

Word vector space for different languages (e.g., English, French) can be meaningfully aligned via a linear transformation [Lample et al’18, Arttextxe et al’18]
Hello, predicting review scores, passively understanding word meaning...

How can machines actively interact with the world, via sequence of intelligent decisions ....?

20th century antecedents

Probability Theory: Betting/gambling games (eg martingales)...
Economics: managing stock portfolios; playing repeated games,..
Control theory for power plants/machines,..
Sequential decision-making: framework (uses: robotics, exploration..)

- Tree of all possible action/interactions. (responses can be stochastic)
- Optimum move = one that minimizes expected loss

Practical difficulty: tree too large to allow full evaluation!

Chess-playing software (circa 1990s): Decision-maker is

1 = LOSE
0 = WIN

list of *handcrafted* rules

+ move evaluation algorithm

(approximate; limited lookahead)
Sequential decision-making (framework)

- Tree of possible action/interactions. (responses can be stochastic)
- Optimum move = one that minimizes expected loss

Practical difficulty: tree too large to allow full evaluation!

Chess-playing software (circa 1990s): Decision-maker is

1 = LOSE
0 = WIN

list of handcrafted rules
+ move evaluation algorithm
(approximate; limited lookahead)
Game-playing via Deep Learning (crude account of Alpha-Go Zero)

Use move evaluation algorithm to evaluate $\ell(\theta) = \text{Pr}[\text{loss}]$

**Estimate gradient of $\ell$ and improve $\theta$!**

Crude method:

$$\ell(\theta + \eta) \approx \ell(\theta) + \eta \nabla \ell \big|_\theta$$

Can “read off” $\nabla \ell \big|_\theta$ using enough $\eta$’s
Part 3

Toward mathematical understanding of Deep Learning
Special case: deep learning (deep = “multilayered”)

\[ f_{\theta}(X_1) \]

Input \( X_1 \) → Matrix \( M_1 \) \( \rightarrow \) Nonlinearity \( \theta: \{ M_1, M_2, \ldots \} \) → Matrix \( M_2 \) \( \rightarrow \) Nonlinearity \( X_2 \) → Matrix \( M_3 \) \( \rightarrow \) Nonlinearity \( X_3 \) → Matrix \( M_4 \) \( \rightarrow \) Nonlinearity \( \ldots \) → Output

“Nonlinearity”: Given a vector, output same vector but negative entries turned to zero.

Training data: \( \{(X_1^{(i)}, Y^{(i)}): i=1,\ldots,N\} \)

\[ \ell(\theta) = \sum_{i=1}^{N} (f_{\theta}(X_1^{(i)}) - Y^{(i)})^2 \]
Some key questions

• Why/when does gradient descent work and how fast? (Nonconvex loss!)

• Why deep (and not shallow)?

• Why doesn’t training overfit to training data? (# parameters >> # training samples). Current deep models capable of achieving zero loss on random data [Zhang et al’17])

• How to interpret the trained model’s inner workings?
Analysis of optimization

Hurdle 1: Nonconvex optimization is NP-hard. No efficient algorithm if $P \neq NP$.

Hurdle 2: Loss $\ell(\theta)$ is essentially a black box to us.

$$\ell(\theta) = \frac{1}{N} \sum_{i=1}^{N} (f_\theta(X^{(i)}) - Y^{(i)})^2$$

Before I work on an object I like it to be well-defined at least...

Lack mathematical description of data $(X^{(i)}, Y^{(i)})$.

“What makes a bunch of pixels an image of a dog?”
Black box analysis (sketch)

- $\nabla \neq 0 \Rightarrow \exists$ descent direction

- But if Hessian ($\nabla^2$) “large”, allows $\nabla$ to fluctuate a lot!

  - To ensure descent, take small enough steps determined by smoothness (norm of $\nabla^2$)

- Guaranteed to get close to point with $\nabla \approx 0$ (speed determined by norms of $\nabla^2$ and $\nabla$). “Stationary point”

More complicated analyses shows Gradient Descent finds “2nd order local minima” [Ge et al.’15]. (Bottom of valley)

(Analysis of optimization algorithms extensively discussed in Michael Jordan’s plenary lecture!)
Analyses of nonconvex opt. : nonblack box

- Phase retrieval, Matrix Completion, Topic Modeling, Sparse coding, Tensor Decomposition, HMM learning,.. Amount to learning special neural nets

"Inverse problems": Assume data generated according to some clean mathematical model. Optimization landscape understood; desired solution is “ground truth” model.

Idea in analysis: Prove direction of movement at current point $\theta$ is positively correlated with desired direction ($\theta - \theta^*$) where $\theta^*$ = global optimum ("Lyapunov function")

[See “Framework for analyzing nonconvex optimization” by A. + Ma, offconvex.org]
Why deep? (Maybe, more expressive? Composition!)

(e.g., If \( p, q \) are cubic polynomials, \( p(q(x)) \) is degree 9.)

[ Eldan-Shamir’16, Telgarsky’17: There exists a function computable with depth \( d+1 \) net of size \( S \) which is not \textit{approximable} by depth \( d \) nets of size \( S^2 \)

Pf Sketch: Characterize max. # of “oscillations” in function computed by depth \( d \) net.

Open: Exhibit above for a natural learning problem.
(Currently out of reach, since we lack good mathematical characterization of “natural”.)
Why deep? (Maybe, helps optimization [A., Cohen, Hazan’18].)

For $l_4$ regression.

$$\ell(\theta) = \sum_{i=1} \left( \langle X^i, \theta \rangle - Y^{(i)} \right)^4$$

$x \rightarrow \langle x, \theta \rangle$

Replace with depth-2 linear deep net

$$\ell(\theta) = \sum_{i=1} \left( \langle X^i, \beta \theta \rangle - Y^{(i)} \right)^4$$

$x \rightarrow \langle x, \theta \rangle \rightarrow \beta \langle x, \theta \rangle$

Unchanged??

No! Gradient Descent changes

$$\theta^{(t+1)} = \theta^{(t)} + \rho^{(t)} \nabla_{\theta^{(t)}} + \sum_{i=1}^{t-1} \mu^{(t,i)} \nabla_{\theta^{(i)}}$$

“Acceleration”? Adaptive learning rate + “memory” of past gradients!

$\mu^{(t,i)} = \mu^{(t,i)}$
Acceleration effect of increasing depth

(UCI regression task...) $l_4$ regression,

- Similar effects observed in nonlinear deep net; eg replace fully connected layer by two layers.
- Some theoretical analysis for multilayer linear nets.
- Proof that acceleration effect due to increase of depth not obtainable via any standard manipulation of original loss function
Why no overfitting? (Even for models with 20M parameters trained on 50k samples)

Popular conjecture: When trained on realistic data, the net’s parameters are constrained ---by problem and/or training ---- to be highly interdependent (e.g., lie on manifold of much lower dimension)

Next few slides: A partial realization of this suggestion

Properly trained nets have “noise stability” property. We prove theorem showing this implies their parameters lie in a lower-dimensional subspace

(“dimension reduction” also appears in Assaf Naor’s plenary lecture!)
Noise stability experiment

[A., Ge, Neyshabur, Zhang ICML'18]

Noise injection: Add \textit{gaussian} $\eta$ to output $x$ of a layer ($|\eta| = |x|$)

Measure percent change in higher layers. (If small, then net is noise stable.)

Results for VGG19 (19 layers)

Key Idea: Can improve reliability of circuits by allowing *redundancy.*

(“noise stability” notion also appears in Gil Kalai’s plenary lecture!)
Noise stability: understanding one layer (no nonlinearity)

\[ \eta : \text{Gaussian noise} \]

\[ \frac{|Mx|}{|x|} \gg \frac{|M\eta|}{|\eta|} \]

\[ \sigma_{\text{max}}(M) \| \left( \sum_i \sigma_i(M)^2 \right)^{1/2} / \sqrt{n} \]

Layer Cushion = ratio ( Roughly speaking.. )

Distribution of singular values in a filter of layer 10 of VGG19. Such matrices are compressible…
Proof sketch: Noise stability $\Rightarrow$ deep net can be made low-dimensional (minimal change to training error)

Idea 1: Compress a layer (randomized; errors introduced are “Gaussian like”)

Idea 2: Errors attenuate as they go through network, due to noise stability. So output changed not much.

Compression:
(1) Generate $k$ random sign matrices $M_1, \ldots, M_k$ (impt: picked before seeing data)
(2) $\hat{A} = \frac{1}{k} \sum_{t=1}^{k} \langle A, M_t \rangle M_t$
Part 4

Taking stock, wrapping up
“Mindless” model-fitting...
None of this is remotely how humans think!

1. Imitation approach has not worked well in the past: airplanes, chess/go etc.

2. Machines’ advantage lies precisely in ability to crunch data.

3. We have little idea at an operational level how humans think.

In fact, machine learning is currently the best hope for figuring out how humans think...
Sample Task: "Decoding" Brain fMRI [Vodrahalli et al, NeuroImage’17]

Not your typical applied math problem (e.g., x-ray tomography). No mathematical model for language!!

Movie scenes

Annotations of movie scenes

Each movie scene paired with text description from external party.

fMRI Machine

fMRI responses

Voxel vector per time step $\in \mathbb{R}^{50,000}$

"Decode??"
Sample Task: “Decoding” Brain fMRI  [Vodrahalli et al, NeuroImage’17]

Movie scenes

Each movie scene paired with text description from external party.

Voxel vector per time step $\in \mathbb{R}^{50,000}$

fMRI responses

fMRI Machine

semantic embeddings of text ("SiF") from [Arora et al’17]

Learnt “Linear transformation”
Hottest trend in academia...

Apply Machine Learning to Discipline X

(X = Physics, Biology, Chemistry, Medicine, Engineering, Neuroscience, Economics/Finance, History, Comparative Literature,....)

Hope mathematics will join in this development!
Concluding thoughts on ML

- Speaks to age-old wonders/mysteries
- New way of looking at the world (via complicated inexact descriptions)
- A new frontier for science and math
- I am optimistic that deep learning methods can be mathematically understood and/or simplified.

THANK YOU!!

(Thanks for feedback: M. Goresky, A. Ionescu, J. Kollar, P. Sarnak, T. Spencer A. Wigderson.)
Formal framework (inherited from classical statistics)

\[ X \xrightarrow{\theta} f_\theta(X) \]

\[ \theta \in \mathbb{R}^d \]

(trainable parameters)

Training data: \( \{(X_1^{(i)}, Y^{(i)}): i=1,..,N\} \)

“Loss”

\[ \ell(\theta) = \sum_{i=1}^{N} (f_\theta(X_1^{(i)}) - Y^{(i)})^2 \]

(Many other loss formulations exist….)

A classical analog: Fourier Transform of periodic \( f: \mathbb{R}^n \to \mathbb{R} \)

\[ f(x) = \sum_{\eta} F(\eta) \exp(\omega \eta \cdot x) \]

“Learning from (x, f(x)) pairs”

\[ F(\eta) = \int f(x) \exp(-\omega \eta \cdot x) dx \]

(l_2 Loss function)
Formal framework (inherited from classical statistics)

\[ X \xrightarrow{\Theta} f_\theta(X) \]

\( \theta \in \mathbb{R}^d \) (trainable parameters)

Training data: \( \{(X_1^{(i)}, Y^{(i)}): i=1, \ldots, N\} \)

“Loss” \( \ell(\theta) = \sum_{i=1}^{N} (f_\theta(X_1^{(i)}) - Y^{(i)})^2 \)

(Many other loss formulations exist….)

A classical analog: Fourier Transform of periodic \( f: \mathbb{R}^n \to \mathbb{R} \)

Practically Infeasible: Needs \( \exp(n) \) samples of \( (x, f(x)) \) for even modest accuracy.

(Real life: \( n = 10^4 \) and \( 10n \) samples.)