Deep Learning

Impressive results

https://deepart.io/
https://deepdreamgenerator.com/

A Neural Algorithm of Artistic Style, Gatys et al, 2015
BUT

It is “alchemy”

- We don’t know why algorithms work or why they don’t (no theory)
- Algorithms are developed through trial and error
- Some results are hard to replicate (many hyperparameters)
- Finding good architectures relies on guesswork
- Very deep networks (more than 40 layers) are difficult to train with backpropagation
- Algorithms are not robust to adversarial examples

AI researchers allege that machine learning is alchemy

By Matthew Hutson | May 3, 2018, 11:15 AM

Ali Rahimi, a researcher in artificial intelligence (AI) at Google in San Francisco, California, took a swipe at his field last December—and received a 40-second ovation for it. Speaking at an AI conference, Rahimi charged that machine learning algorithms, in which computers learn through trial and error, have become a form of “alchemy.” Researchers, he said, do not know why some algorithms work and others don’t, nor do they have rigorous criteria for choosing one AI architecture over another. Now, in a paper presented on 30 April at the International Conference on Learning Representations in Vancouver, Canada, Rahimi and his collaborators document examples of what they see as the alchemy problem and offer prescriptions for bolstering AI’s rigor.

“There’s an anguish in the field,” Rahimi says. “Many of us feel like we’re operating on an alien technology.”

“Machine learning has become alchemy”
Ali Rahimi
NIPS 2017 Test of Time Award

Science Mag, May 2018
Questions

Can the interface between NA and Game theory offer some insights?

Is there an approach that

- Is amenable to some degree of analysis?
- Produces a network without guesswork?
  (plug and play, no tweaking of hyperparameters, no guessing of the architecture)
- Enables the training of very deep networks?
  (50,000 layers or more) and the exploration of their properties
- Provides some insight on developing a rigorous theory for deep learning?

Initial results

Interface between Game Theory and NA

Deep Learning

Learning is solving an interpolation problem

\[ \mathbf{x} \xrightarrow{u} \mathbf{y} \]

\( u : \text{Unknown} \)

Given \( y_i = u(x_i) \) for \( i = 1, \ldots, N \), approximate \( u \)
Given kernel $K$ approximate $u(x)$ with

$$v(x) = \sum_i c_i K(x_i, x)$$

$c$ such that $v(x_i) = y_i$ for all $i$
BUT

- What if $N$ is large?
- Which kernel do we pick?
A kernel $K$ is good if the number of interpolation points can be halved without significant loss in accuracy.

$v$: Interpolate with $K$ and $N$ points

$w$: Interpolate with $K$ and $N/2$ points

$$
\rho = \frac{||v-w||^2}{||v||^2}
$$

$$
||v||^2 = \sup_\phi \frac{\left(\int \phi(x)v(x) \, dx\right)^2}{\int \phi(x)K(x,x')\phi(x') \, dx \, dx'}
$$

Good kernel $\iff$ Small $\rho$
Kernel Flow

Learns kernels of the form

\[ K_n(x, x') = K_1(F_n(x), F_n(x')) \]

\(K_1:\) kernel (e.g. \(K_1(x, x') = e^{-\frac{|x-x'|^2}{\gamma^2}}\))

\(F_n:\) Flow in input space

\[ F_n : X \rightarrow X \]

\(F_1 = I_d\)

\[ F_n \quad \rightarrow \quad F_{n+1} \]

Data
Step $n \rightarrow n + 1$

Assume $F_n$ known

Images of the $N$ training points under $F_n$
Select $N_f$ at random out of $N$
Select $N_f/2$ at random out of $N_f$
Player I
Selects the values/labels of the blue points $F'_n(x_i)$ to be $y_i$ (training labels)

Player II
Sees values/labels $y_i$ of the $N_c = N_f/2$ green points must predict the values of the blue points

$\rho$: Relative error in $\| \cdot \|$ norm
$\| \cdot \|$: RKHS norm associated with $K_1$
Move the $N_f$ points in the gradient descent direction of $\rho$
Rig the game in favor of Player II

Move the $N_f$ points in the gradient descent direction of $\rho$
Move the remaining $N - N_f$ points via interpolation with kernel $K_1$

Move any point $x$
via interpolation with kernel $K_1$ \( F_{n+1} \)
Repeat

$F_{n+1}$ known

Images of the $N$ training points under $F_{n+1}$
Kernel Flow

Produces a deep hierarchical kernel of the form

\[ K_n(x, x') = K_{n-1}(x + \epsilon \sum_{i=1}^{N_f} c_i K_{n-1}(x_{\sigma_f(i)}, x), x', \epsilon \sum_{i=1}^{N_f} c_i K_{n-1}(x_{\sigma_f(i)}, x')) \]

and a flow of the form

\[ F_{n+1} = (I_d + \epsilon G_{n+1}) \circ F_n \]

\[ G_{n+1}(x) = \sum_{i=1}^{N_f} c_i K_1(F_n(x_{\sigma_f(i)}), x) \]

Identified as the steepest gradient descent direction of \( \rho \).
Application: Swiss Roll Cheesecake

\[ N = 100 \text{ data points } x_i \in \mathbb{R}^2 \]
\[ y_i = -1 \text{ if point at } x_i \text{ is red} \]
\[ y_i = +1 \text{ if point at } x_i \text{ is blue} \]

Objective:
Visualize \( n \to F_n(x_i) \)
\[ F_n(x_i) \quad \text{Gaussian Kernel, } N_f = N \]
$F_n(x_i)$  \hspace{1cm} \text{Gaussian Kernel, } N_f = N, \text{ large } \epsilon
Application to Fashion-MNIST

\[ N = 60000 \]
\[ N_f = 600 \]
12000 layers, large steps
50000 layers, small steps
Application to MNIST

\[ N = 60000 \]
\[ N_f = 600 \]
12000 layers
Average distance, inter-class $y_i \neq y_j$

$$\mathbb{E} \left[ \left| F_n(x_i) - F_n(x_j) \right|^2 \right]$$
Average distance, in-class

\[ y_i = y_j \]

\[ \mathbb{E} \left[ \left| F_n(x_i) - F_n(x_j) \right|^2 \right] \]
Ratio average distances inter/in
Classify 10000 test points

Use kernel $K_n$ and $N_I$ interpolation points selected at random

$N_I = 6000, 600, 60, 10$

$N_I = 10 \quad \leftrightarrow \quad \text{Interpolate with only 1 point per class}$
Fashion-MNIST Test Error vs layer

Error vs Layer 10 interpolation points
Fashion-MNIST Test Error vs layer
Fashion-MNIST Test Error vs layer

Error vs Layer 600 interpolation points

0.25
0.2
0.15
0.1
0.05
0.0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
×10^4
Fashion MNIST

For $15000 \leq n \leq 25000$

9.7% average error with $K_n$ and 600 interpolation points

<table>
<thead>
<tr>
<th>$N_I$</th>
<th>Average error</th>
<th>Min error</th>
<th>Max error</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6000</td>
<td>0.0969</td>
<td>0.0944</td>
<td>0.1</td>
<td>$7.56 \times 10^{-4}$</td>
</tr>
<tr>
<td>600</td>
<td>0.0977</td>
<td>0.0951</td>
<td>0.101</td>
<td>$8.57 \times 10^{-4}$</td>
</tr>
<tr>
<td>60</td>
<td>0.114</td>
<td>0.0958</td>
<td>0.22</td>
<td>0.0169</td>
</tr>
<tr>
<td>10</td>
<td>0.444</td>
<td>0.15</td>
<td>0.722</td>
<td>0.096</td>
</tr>
</tbody>
</table>

For $49900 \leq n \leq 50000$

10% average error with $K_n$ and 10 interpolation points

<table>
<thead>
<tr>
<th>$N_I$</th>
<th>Average error</th>
<th>Min error</th>
<th>Max error</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6000</td>
<td>0.10023</td>
<td>0.0999</td>
<td>0.1006</td>
<td>$1.6316 \times 10^{-4}$</td>
</tr>
<tr>
<td>600</td>
<td>0.10013</td>
<td>0.0999</td>
<td>0.1004</td>
<td>$1.1671 \times 10^{-4}$</td>
</tr>
<tr>
<td>60</td>
<td>0.10018</td>
<td>0.0999</td>
<td>0.1005</td>
<td>$1.445 \times 10^{-4}$</td>
</tr>
<tr>
<td>10</td>
<td>0.10018</td>
<td>0.0996</td>
<td>0.1009</td>
<td>$2.2941 \times 10^{-4}$</td>
</tr>
</tbody>
</table>
MNIST Test Error vs layer

Error vs Layer 10 interpolation points
MNIST Test Error vs layer

Error vs Layer 60 interpolation points
MNIST Test Error vs layer

Error vs Layer 6000 interpolation points
$N = 60000$

10000 test points

$N_f = 600$

$n = 12000$

1.5% average error with $K_n$ and 10 interpolation points

<table>
<thead>
<tr>
<th>$N_I$</th>
<th>Average error</th>
<th>Min error</th>
<th>Max error</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6000</td>
<td>0.014</td>
<td>0.0136</td>
<td>0.0143</td>
<td>$1.44 \times 10^{-4}$</td>
</tr>
<tr>
<td>600</td>
<td>0.014</td>
<td>0.0137</td>
<td>0.0142</td>
<td>$9.79 \times 10^{-5}$</td>
</tr>
<tr>
<td>60</td>
<td>0.0141</td>
<td>0.0136</td>
<td>0.0146</td>
<td>$2.03 \times 10^{-4}$</td>
</tr>
<tr>
<td>10</td>
<td>0.015</td>
<td>0.0136</td>
<td>0.0177</td>
<td>$7.13 \times 10^{-4}$</td>
</tr>
</tbody>
</table>
$F_n(x_i)$, Gaussian kernel + nugget, $\epsilon = 0.2$
Instantaneous velocity field

\[ F_{n+1}(x) - F_n(x) \]
Average velocity field

\[10\left(F_{n+300}(x) - F_n(x)\right)/300\]
The effective dynamical system

As $\epsilon \downarrow 0$, $F_{\text{round}}(\frac{t}{\epsilon})(x) \rightarrow F(t, x)$

$$\frac{\partial F(t, x)}{\partial t} = -\mathbb{E}_{X, \pi} \left[ \left( \nabla_Z \rho(X, Z, \pi) \right)^T \left( K_1(Z, Z) \right)^{-1} K_1(Z, x) \right]_{Z=F(X,t)}$$

$$\rho(X, Z, \pi) = 1 - \frac{u(X)^T \pi^T K_1(\pi Z, \pi Z)^{-1} \pi u(X)}{u(X)^T (K_1(Z, Z))^{-1} u(X)}$$

- $X$: random vector of $\mathcal{X}^{N_f}$ representing the random sampling of the training data in a batch size $N_f$
- $u(X) \in \mathcal{Y}^{N_f}$ is the vector whose entries are the labels of the entries of $X \in \mathcal{X}^{N_f}$
- $\pi$: Random $N_c \times N_f$ matrix corresponding to the selection of $N_c$ elements at out $N_f$ (at random, uniformly, without replacement)
Thank you