MACHINE LEARNING, NUCLEAR PHYSICS, AND ALGORITHM DEVELOPMENT FOR DATA ANALYSIS IN NUCLEAR RESEARCH

MICHELLE KUCHERA
DAVIDSON COLLEGE

IAEA WORKSHOP ON COMPUTATIONAL NUCLEAR SCIENCE AND ENGINEERING
16 JULY 2021
MACHINE LEARNING, NUCLEAR PHYSICS, AND ALGORITHM DEVELOPMENT FOR DATA ANALYSIS IN NUCLEAR RESEARCH

MICHHELLE KUCHERA
DAVIDSON COLLEGE

IAEA WORKSHOP ON COMPUTATIONAL NUCLEAR SCIENCE AND ENGINEERING
16 JULY 2021
MACHINE LEARNING IN FUNDAMENTAL NUCLEAR AND PARTICLE PHYSICS

MICHELLE KUCHERA
B.S., M.S. PHYSICS
M.S., PH.D. COMPUTATIONAL SCIENCE

\[ w_1 = w_1 + \eta * \frac{\partial f}{\partial q_1} \frac{\partial q_1}{\partial w_1} \]
\[ q_1 = x_1 w_1 \]
\[ w_2 = w_2 + \eta * \frac{\partial f}{\partial q_1} \frac{\partial q_2}{\partial w_2} \]
\[ q_2 = x_2 w_2 \]
\[ f = q_1 + q_2 \]
\[ \frac{\partial f}{\partial q_1} = 1 \]
\[ \frac{\partial f}{\partial q_2} = 1 \]
\[ J(w) = f - \hat{f} \]
EXPERIMENTAL DATA

EXPERIMENTAL DATA

FRIB

AT-TPC

Jefferson Lab

CLAS 12

CERN

CMS
MACHINE LEARNING: LEARNING FROM DATA

Without Machine Learning

With Machine Learning

*VERY SPECIFIC INSTRUCTIONS*
Approximation capabilities of multilayer feedforward networks

Kurt Hornik

Abstract

We show that standard multilayer feedforward networks with as few as a single hidden layer and arbitrary bounded and nonconstant activation function are universal approximators with respect to $L^1(\mu)$ performance criteria, for arbitrary finite input environment measures $\mu$, provided only that sufficiently many hidden units are available. If the activation function is continuous, bounded and nonconstant, then continuous mappings can be learned uniformly over compact input sets. We also give very general conditions ensuring that networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives.
MATHEMATICS
\[ \hat{f} = x_1 w_1 + x_2 w_2 \]
REGRESSION

\[ J(w) = f - \hat{f} \]

\[ \hat{f} = x_1 w_1 + x_2 w_2 \]

Diagram:

- \( x_1 \)
- \( w_1 \)
- \( q_1 = x_1 w_1 \)
- \( x_2 \)
- \( w_2 \)
- \( q_2 = x_2 w_2 \)
- \( \hat{f} = q_1 + q_2 \)

Loss function:
SUPERVISED LEARNING

$J(w) = f - \hat{f}$

$\hat{f} = x_1 w_1 + x_2 w_2$

$q_1 = x_1 w_1$

$q_2 = x_2 w_2$

$\hat{f} = q_1 + q_2$
\[ w_1 = w_1 + \eta^* \frac{\partial J}{\partial f} \frac{\partial f}{\partial q_1} \frac{\partial q_1}{\partial w_1} \]

\[ w_2 = w_2 + \eta^* \frac{\partial J}{\partial f} \frac{\partial f}{\partial q_2} \frac{\partial q_2}{\partial w_2} \]

\[ \frac{\partial q_1}{\partial w_1} = x_1 \]

\[ \frac{\partial q_1}{\partial q_1} = 1 \]

\[ \frac{\partial f}{\partial q_1} = 1 \]

\[ \frac{\partial f}{\partial q_2} = 1 \]

\[ \frac{\partial f}{\partial w_1} = x_1 \]

\[ \frac{\partial f}{\partial w_2} = x_2 \]

Loss function

\[ J(w) = f - \hat{f} \]
LOGISTIC REGRESSION

\[
f = \frac{1}{1 + e^{-(x_1w_1 + x_2w_2)}}
\]
LOGISTIC REGRESSION
\[ f = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2)}} \]
Features $x_1$, $x_2$, $x_3$, $x_4$ connected to weights $w_1$, $w_2$, $w_3$, $w_4$. The weights are summed and passed through a nonlinearity $N$, resulting in an output.
BACKPROPAGATION

\[ w_1 = w_1 + \eta \frac{\partial J}{\partial f} \frac{\partial f}{\partial q_1} \frac{\partial q_1}{\partial w_1} \]

\[ \frac{\partial q_1}{\partial w_1} = x_1 \]

\[ q_1 = x_1w_1 \]

\[ \frac{\partial f}{\partial q_1} = 1 \]

\[ f = q_1 + q_2 \]

\[ \frac{\partial f}{\partial q_2} = 1 \]

\[ w_2 = w_2 + \eta \frac{\partial J}{\partial f} \frac{\partial f}{\partial q_2} \frac{\partial q_2}{\partial w_2} \]

\[ \frac{\partial q_2}{\partial w_2} = x_2 \]

Loss function

\[ J(w) = f - \hat{f} \]
AUTOMATIC DIFFERENTIATION

TensorFlow

PyTorch
MACHINE LEARNING:
LEARNING FROM DATA

Without Machine Learning

With Machine Learning

VERY SPECIFIC INSTRUCTIONS

DATA
\[ \hat{f} = x_1 w_1 + x_2 w_2 \]
MACHINE LEARNING
MACHINE LEARNING

SUPERVISED LEARNING
\[ w_1 = w_1 + \eta \star \frac{\partial f}{\partial q_1} \frac{\partial q_1}{\partial w_1} \]

\[ w_2 = w_2 + \eta \star \frac{\partial f}{\partial q_1} \frac{\partial q_1}{\partial w_2} \]

\[ q_1 = x_1 w_1 \]

\[ q_2 = x_2 w_2 \]

\[ f = q_1 + q_2 \]

\[ \frac{\partial f}{\partial q_1} = 1 \]

\[ \frac{\partial f}{\partial q_2} = 1 \]

\[ \frac{\partial J}{\partial f} = 1 \]

Loss function

\[ J(w) = f - \hat{f} \]
LOGISTIC REGRESSION

\[ f = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2)}} \]
LOGISTIC REGRESSION
**LOGISTIC REGRESSION**

\[
f = \frac{1}{1 + e^{-(x_1w_1 + x_2w_2)}}
\]
Features $x_1, x_2, x_3, x_4$ weighted by $w_1, w_2, w_3, w_4$ summed and passed through a nonlinearity to produce the output.
Application 1: How can experimental observables constrain theoretical models?
THEORY ↔ EXPERIMENT

Input ➔ Output

Training

Parameter Space

Theory

Observable Space

Observable Space

Neural Nets

Parameter Space

Application

Experimental Data

Neural Nets

Visualization

N. SATO, JEFFERSON LAB
THEORY ↔ EXPERIMENT

Training
Parameter Space
Observable Space
Observables

Application
Experimental Data
Neural Nets
Visualization

INPUT → OUTPUT
MIXTURE DENSITY NETWORK

\[ \text{MDN} \]

\[ \sigma_1, \sigma_2, \sigma_3, \sigma_{P-1} \]

Figure 2: Architecture of the kinematics-independent inverse mapper.

Output Layer Interpretation:

\[ p(t|x) = \sum_{k=1}^{K} \pi_k(x) N \left( t|\mu_k(x), \sigma_k^2(x) \right) \]
FAST MAPPING TO THEORETICAL PARAMETERS

Bayesian Neural Networks

Training — Bayesian inference

Can we make predictions with accurate error estimates?

$pMSSM$ parameters $\rightarrow$ total SUSY cross section
FAST MAPPING TO THEORETICAL PARAMETERS

16 million times faster

than theory codes!


https://alpha-davidson.github.io/TensorBNN
CONVOLUTIONAL NEURAL NETWORKS

CLASSIFICATION
CONVOLUTIONAL NEURAL NETWORKS
CONVOLUTIONAL NEURAL NETWORKS

Feature Extraction

Latent Space

Task
DISCRETE CONVOLUTION

Input

Filter

Feature Map

ADAPTED FROM DEEP LEARNING, ADAM GIBSON & JOSH PATTERSON
CONVOLUTIONAL NEURAL NETWORKS
CONVOLUTIONAL NEURAL NETWORKS
CONVOLUTIONAL NEURAL NETWORKS
MAX POOLING

max pool with 2x2 filters and stride 2

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>
CONVOLUTIONAL NEURAL NETWORKS
“GoogLeNet network with all the bells and whistles”
CHOOSING AN ARCHITECTURE

HOW MANY LAYERS?
HOW MANY NODES PER LAYER?
LEARNING RATE
DROPOUT?
WHAT ACTIVATION FUNCTION(S)?
HOW MANY CONVOLUTION LAYERS?
FILTER SIZE?
STRIDE?
POOLING?
PRE-TRAINED MODELS

Feature Extraction

Latent Space

Task
PRE-TRAINED MODELS
PRETRAINED MODELS

ALEXNET

VGG

RESNETS

INCEPTION

XCEPTION
Application 2: Can we use machine learning to accurately classify events in detectors?

Metrics
Detect Lung Cancer

99% Accuracy
<table>
<thead>
<tr>
<th></th>
<th>Proton</th>
<th>Not Proton</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE</strong></td>
<td>TP (TP)</td>
<td>FN (FN)</td>
</tr>
<tr>
<td><strong>FALSE</strong></td>
<td>FP (FP)</td>
<td>TN (TN)</td>
</tr>
</tbody>
</table>

**PREDICTED**

**Proton**

**Not Proton**

**TP** (TRUE POSITIVE)

**FN** (FALSE NEGATIVE)

**FP** (FALSE POSITIVE)

**TN** (TRUE NEGATIVE)
\[ \text{precision} = \frac{TP}{TP + FP} \]
\[ \text{recall} = \frac{TP}{TP + FN} \]
\[ \text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \]
\[ \text{accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \]
A confusion matrix for a perfect model showing:

- True Positive (TP)
- False Negative (FN)
- False Positive (FP)
- True Negative (TN)
Application 2: Can we use machine learning to accurately classify events in detectors?
ACTIVE-TARGET TIME PROJECTION CHAMBER (AT-TPC)

Quantum energy states of potential well including angular momentum effects.

Further splitting from spin-orbit effect

Multiplicity of states

Closed shells indicated by "magic numbers" of nucleons.

Number of neutrons (N)

Z=20 Calcium

N=28

Lead Z=82

N=126
EXPERIMENTAL DATA

AT-TPC

HALL B
VGG16 ARCHITECTURE

PRE-TRAINED ON IMAGENET DATA!
<table>
<thead>
<tr>
<th>Experiment</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental → Experimental</td>
<td>0.96</td>
<td>0.90</td>
<td>0.93</td>
</tr>
<tr>
<td>Simulated → Simulated</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Simulated → Experimental</td>
<td>0.90</td>
<td>0.60</td>
<td>0.72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>0.93</td>
<td>0.95</td>
</tr>
<tr>
<td>Experiment</td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>Experimental → Experimental</td>
<td>0.96</td>
<td>0.90</td>
</tr>
<tr>
<td>Simulated → Simulated</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Simulated → Experimental</td>
<td>0.90</td>
<td>0.60</td>
</tr>
</tbody>
</table>

**AT-TPC**

**HALL B**

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97</td>
<td>0.93</td>
<td>0.95</td>
</tr>
</tbody>
</table>

6x faster!
MACHINE LEARNING

UNSUPERVISED LEARNING
CLUSTERING — KMEANS

Goal: minimize pairwise distances between points in same cluster

$$\min \sum_{i=1}^{k} \frac{1}{2N} \sum_{x,y,x \neq y} (\vec{x} - \vec{y})^2$$

Goal: maximize pairwise distances between points in different clusters
CLUSTERING — KMEANS
<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>10000000</td>
</tr>
<tr>
<td>01000000</td>
<td>01000000</td>
</tr>
<tr>
<td>00100000</td>
<td>00100000</td>
</tr>
<tr>
<td>00010000</td>
<td>00010000</td>
</tr>
<tr>
<td>00001000</td>
<td>00001000</td>
</tr>
<tr>
<td>00000100</td>
<td>00000100</td>
</tr>
<tr>
<td>Input</td>
<td>Output</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>10000000</td>
<td>10000000</td>
</tr>
<tr>
<td>01000000</td>
<td>01000000</td>
</tr>
<tr>
<td>00100000</td>
<td>00100000</td>
</tr>
<tr>
<td>00010000</td>
<td>00010000</td>
</tr>
<tr>
<td>00001000</td>
<td>00001000</td>
</tr>
<tr>
<td>00000100</td>
<td>00000100</td>
</tr>
<tr>
<td>Input</td>
<td>A1</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>10000000</td>
<td>0.9911</td>
</tr>
<tr>
<td>01000000</td>
<td>0.9892</td>
</tr>
<tr>
<td>00100000</td>
<td>0.0094</td>
</tr>
<tr>
<td>00010000</td>
<td>0.9840</td>
</tr>
<tr>
<td>00001000</td>
<td>0.0139</td>
</tr>
<tr>
<td>00000100</td>
<td>0.0128</td>
</tr>
</tbody>
</table>
Learning of the encoding for input 00000010
GENERATIVE MODELS

MICHELLE KUCHERA
DAVIDSON COLLEGE

ECT* TALENT SUMMER SCHOOL
02 JULY 2020
How do we know that we are providing a latent vector that represents those seen in training?

Variational Autoencoder
Encode to two outputs for each latent dimension: mean and std dev

Sample similar points in latent space, decode, and compare with regularization
https://blog.keras.io/building-autoencoders-in-keras.html
GENERATIVE MODELS

SIMULATION
GENERATIVE ADVERSARIAL NETWORKS (GANS)
maximize $D(G(z))$

minimize $D(G(z))$
GAN (DCGAN)

\[ \nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} \log D(x^{(i)}) + \log \left(1 - D\left(G(z^{(i)})\right)\right) \]

\[ \nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G(z^{(i)})\right)\right) \text{ or } \nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} -\log \left(D\left(G(z^{(i)})\right)\right) \]

Real image \( x \)

\[ z \sim \mathcal{N}(0, 1) \text{ or } z \sim \mathcal{U}(-1, 1) \]

Generator

Discriminator

Cost

WGAN

\[ \nabla_{\omega} \left[ \frac{1}{m} \sum_{i=1}^{m} f_{\omega}(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_{\omega}(g_{\theta}(z^{(i)})) \right] \]

Real image \( x \)

\[ z \sim \mathcal{N}(0, 1) \text{ or } z \sim \mathcal{U}(-1, 1) \]

Generator

Critic

Cost

\[ -\nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} f_{\omega}(g_{\theta}(z^{(i)})) \]
Application 3: Can we use machine learning to simulate data?
CONDITIONAL GAN

Accepted. Computational High Energy and Nuclear Physics (2021)
Total Distributions

Conditional GAN

Conditional Jet Distributions

Conditional Wasserstein Generative Adversarial Networks for Fast Detector Simulation. John Blue, Braden Kronheim, Michelle Kuchera, Raghuram Ramanujan

Accepted. Computational High Energy and Nuclear Physics (2021)
**CONDITIONAL GAN**

Conditional Distributions

---

EXAMPLE WORKFLOW

DATA

PRE-

FINE

TRAIN

MODIFY

BENCHMARK!

LOGISTIC REGRESSION

SUPPORT VECTOR MACHINES

NEURAL NETWORKS
ACKNOWLEDGMENTS

• Raghu Ramanujan, Meg Houck, Eleni Tsitinidi, Jose Cruz, Andrew Hoyle, Michael Robertson, Evan Pritchard, Robert Solli, John Blue, Zach Nussbaum, Ryan Strauss, Jack Taylor

• JLab/ODU - Theory Center, Hall B: Nobuo Sato, Wally Melnitchouk, Yaohang Li, Yasir Alanazi, Manal Almaeen, Gagik Gavalian

• NSCL/FRB - AT-TPC Group