Machine Learning Overview

From artificial intelligence to physics

Stefano Carrazza

23rd Workshop on Electronic Excitations,
10-14 September 2018, University of Milan

European Organization for Nuclear Research (CERN)

Acknowledgement: This project has received funding from HICCUP ERC Consolidator grant (614577) and by the European Unions Horizon 2020 research and innovation programme under grant agreement no. 740006.
Why talk about machine learning?
Why talk about machine learning?

because

- it is an essential set of algorithms for building models in science,
- fast development of new tools and algorithms in the past years,
- nowadays it is a requirement in experimental and theoretical physics,
- large interest from the HEP community: IML, conferences, grants.
Topics:

- A.I. and M.L. overview
- Non-linear models
- From physics to ML
Artificial Intelligence
ARTIFICIAL INTELLIGENCE

MACHINE LEARNING

DEEP LEARNING

Turing Test Devised 1950
ELIZA 1964 - 1966
Edward Shortliffe writes MYCIN, an Expert or Rule based System, to classify blood disease 1970s
IBM Deep Blue defeats Grand Master Garry Kasparov in chess 1996
ImageNet Feeds Deep Learning 2009
AlphaGo defeats Go champion Lee Sedol 2016
Artificial intelligence (A.I.) is the science and engineering of making intelligent machines. (John McCarthy ‘56)
Artificial intelligence (A.I.) is the science and engineering of making intelligent machines. (John McCarthy ‘56)

A.I. consist in the development of computer systems to perform tasks commonly associated with intelligence, such as learning.
There are two categories of A.I. tasks:

- **abstract and formal:** easy for computers but difficult for humans, e.g. play chess (IBM’s Deep Blue 1997).
  → *Knowledge-based* approach to artificial intelligence.

- **intuitive for humans but hard to describe formally:** e.g. recognizing faces in images or spoken words.
There are two categories of A.I. tasks:

- **abstract and formal**: easy for computers but difficult for humans, e.g. play chess (IBM’s Deep Blue 1997).
  → *Knowledge-based* approach to artificial intelligence.

- **intuitive for humans but hard to describe formally**: e.g. recognizing faces in images or spoken words.
  → *Concept* capture and generalization
A.I. technologies

Historically, the *knowledge-based* approach has not led to a major success with intuitive tasks for humans, because:

- requires human *supervision* and hard-coded *logical inference rules*.
- lacks of *representation learning* ability.
Historically, the *knowledge-based* approach has not led to a major success with intuitive tasks for humans, because:

- requires human *supervision* and hard-coded *logical inference rules*.
- lacks of *representation learning* ability.

**Solution:**

The A.I. system needs to *acquire its own knowledge*. This capability is known as *machine learning* (ML).

→ *e.g.* write a program which learns the task.
Machine Learning
Definition from A. Samuel in 1959:
Field of study that gives computers the ability to learn without being explicitly programmed.
**Definition from A. Samuel in 1959:**
Field of study that gives computers the ability to learn without being explicitly programmed.

**Definition from T. Mitchell in 1998:**
A computer program is said to *learn* from experience $E$ with respect to some class of *tasks* $T$ and *performance measure* $P$, if its performance on $T$, as measured by $P$, improves with experience $E$. 
ML applications in our “day life”
Machine learning examples

Thanks to work in A.I. and new capability for computers:

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records
Machine learning examples

Thanks to work in A.I. and new capability for computers:

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records

- **Intuitive tasks for humans:**
  - Autonomous driving
  - Natural language processing
  - Robotics (reinforcement learning)
  - Game playing (DQN algorithms)
Machine learning examples

Thanks to work in A.I. and new capability for computers:

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records

- **Intuitive tasks for humans:**
  - Autonomous driving
  - Natural language processing
  - Robotics (reinforcement learning)
  - Game playing (DQN algorithms)
Machine learning examples

Thanks to work in A.I. and new capability for computers:

- **Database mining:**
  - Search engines
  - Spam filters
  - Medical and biological records

- **Intuitive tasks for humans:**
  - Autonomous driving
  - Natural language processing
  - Robotics (reinforcement learning)
  - Game playing (DQN algorithms)

- **Human learning:**
  - Concept/human recognition
  - Computer vision
  - Product recommendation
ML applications in condensed matter
ML in condensed matter

Some recent examples:

- **Phase transitions and classification**: unsupervised learning.
  
  Lei Wang,  

  J. Carrasquilla and R. Melko  

  E.P. van Nieuwenburg, Y. Liu, S. Huber  
Some recent examples:

- **Phase transitions and classification**: unsupervised learning.
- **State compression and representation**: reinforcement learning.

ML in condensed matter

Some recent examples:

- **Phase transitions and classification**: unsupervised learning.
- **State compression and representation**: reinforcement learning.
- **Experimental / numerical protocols**: neural networks.


ML in condensed matter

Some recent examples:

- **Phase transitions and classification**: unsupervised learning.
- **State compression and representation**: reinforcement learning.
- **Experimental / numerical protocols**: neural networks.
- **Physics → ML**: RTBM, Tensor Networks.
ML applications in HEP
There are many applications in experimental HEP involving the **LHC measurements**, including the **Higgs discovery**, such as:

- Tracking
- Fast Simulation
- Particle identification
- Event filtering
ML in experimental HEP

Some remarkable examples are:

- **Signal-background detection:**
  Decision trees, artificial neural networks, support vector machines.

- **Jet discrimination:**

- **HEP detector simulation:**
  Generative adversarial networks, e.g. LAGAN and CaloGAN.
ML in theoretical HEP

- **Supervised learning:**
  - The structure of the proton at the LHC
    - parton distribution functions
  - Theoretical prediction and combination
  - Monte Carlo reweighting techniques
    - neural network Sudakov
  - BSM searches and exclusion limits

- **Unsupervised learning:**
  - Clustering and compression
    - PDF4LHC15 recommendation
  - Density estimation and anomaly detection
    - Monte Carlo sampling
Machine learning algorithms:

- **Supervised learning:**
  - regression, classification, ...

Supervised learning:

- Input Data
- Processing
- Output
- Algorithm
- Supervisor
- Training Data Set
- Desired Output
- Algorithm
- Processing
- Output
Machine learning algorithms:

- **Supervised learning:** regression, classification, ...
- **Unsupervised learning:** clustering, dim-reduction, ...

Unsupervised learning:

Input Data → Processing → Algorithm → No Training Data Set → Unknown Output → Discover Interpretation from Features → Output
Machine learning algorithms:

- **Supervised learning:**
  regression, classification, ...

- **Unsupervised learning:**
  clustering, dim-reduction, ...

- **Reinforcement learning:**
  real-time decisions, ...

\[
\text{Reinforcement learning}
\]

Input Data → Algorithm → Agent → Environment → Best Action → Reward → Algorithm

Output

16
Machine learning algorithms

- Deep Boltzmann Machine (DBM)
- Deep Belief Networks (DBN)
- Convolutional Neural Network (CNN)
- Stacked Auto-Encoders
- Random Forest
- Gradient Boosting Machines (GBM)
- Boosting
- Bootstrapped Aggregation (Bagging)
- AdaBoost
- Stacked Generalization (Blending)
- Gradient Boosted Regression Trees (GBRT)
- Radial Basis Function Network (RBFN)
- Perceptron
- Back-Propagation
- Hopfield Network
- Ridge Regression
- Least Absolute Shrinkage and Selection Operator (LASSO)
- Elastic Net
- Least Angle Regression (LARS)
- Cubist
- One Rule (OneR)
- Zero Rule (ZeroR)
- Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
- Linear Regression
- Ordinary Least Squares Regression (OLSR)
- Stepwise Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Scatterplot Smoothing (LOESS)
- Logistic Regression
- Naive Bayes
- Averaged One-Dependence Estimators (AODE)
- Bayesian Belief Network (BBN)
- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Bayesian Network (BN)
- Classification and Regression Tree (CART)
- Iterative Dichotomiser 3 (ID3)
- C4.5
- C5.0
- Chi-squared Automatic Interaction Detection (CHAID)
- Decision Stump
- Conditional Decision Trees
- Principal Component Analysis (PCA)
- Partial Least Squares Regression (PLSR)
- Sammon Mapping
- Multidimensional Scaling (MDS)
- Projection Pursuit
- Principal Component Regression (PCR)
- Partial Least Squares Discriminant Analysis
- Mixture Discriminant Analysis (MDA)
- Quadratic Discriminant Analysis (QDA)
- Regularized Discriminant Analysis (RDA)
- Flexible Discriminant Analysis (FDA)
- Linear Discriminant Analysis (LDA)
- k-Nearest Neighbour (kNN)
- Learning Vector Quantization (LVQ)
- Self-Organizing Map (SOM)
- Locally Weighted Learning (LWL)
- k-Means
- k-Medians
- Expectation Maximization
- Hierarchical Clustering

More than 60 algorithms.
The operative workflow in ML is summarized by the following steps:

The best model is then used to:

- supervised learning: make predictions for new observed data.
- unsupervised learning: extract features from the input data.
However, the selection of the appropriate model comes with trade-offs:

- **Prediction accuracy vs interpretability:**
  - e.g. linear model vs splines or neural networks.
Model representation trade-offs

However, the selection of the appropriate model comes with trade-offs:

- **Prediction accuracy vs interpretability:**
  → e.g. linear model vs splines or neural networks.

- **Optimal capacity/flexibility:** number of parameters, architecture
  → deal with overfitting, and underfitting situations
Perform hyperparameter tune coupled to cross-validation:

- **Grid/random search**
  - *Run I*: Cross-validation, Test set
  - *Run II*: Cross-validation, Test set
  - *Run n*: Cross-validation, Test set

Easy parallelization at search and cross-validation stages.
Artificial neural networks
Limitations of linear models

Why not linear models everywhere?

Example: consider 1 image from the MNIST database:
Each image has 28x28 pixels = 785 features (x3 if including RGB colors).
If consider quadratic function $O(n^2)$, so linear models are impractical.
Solution: use non-linear models.
Limitations of linear models

Why not linear models everywhere?

**Example:** consider 1 image from the MNIST database:

Each image has 28x28 pixels = 785 features (x3 if including RGB colors).

If consider quadratic function $O(n^2)$ so linear models are impractical.
Limitations of linear models

Why not linear models everywhere?

**Example:** consider 1 image from the MNIST database:

Each image has 28x28 pixels = 785 features (x3 if including RGB colors). If consider quadratic function $O(n^2)$ so linear models are impractical.

**Solution:** use non-linear models.
Non-linear models timeline

1943: Neural Nets
1958: Perceptron
1974: Backpropagation
1980: Neocogitron, SOMs
1985: Boltzmann Machine
1986: Multilayer Perceptron, Restricted BMs, RNNs
1982: Hopfield Networks
1990: LeNet
1997: LSTMs, BRNNs
2006: Deep BMs, Deep Belief NNs
2012: Dropout
2014: GANs
2017: RTBMs
2020:
Artificial neural networks are computer systems inspired by the biological neural networks in the brain.

Currently the state-of-the-art technique for several ML applications.
We can imagine the following data communication pattern:
Neuron model

Schematically:

where

- each **node** has an associate weights and bias $w$ and inputs $x$,
- the output is modulated by an activation function, $g$.

Some examples of activation functions: sigmoid, tanh, linear, ...

$$g_w(x) = \frac{1}{1 + e^{-w^T x}}, \quad \tanh(w^T x), \quad x.$$
In practice, we simplify the bias term with $x_0 = 1$.

Neural network → connecting multiple units together.

where

- $a_i^{(l)}$ is the activation of unit $i$ in layer $l$,
- $w_{ij}^{(l)}$ is the weight between nodes $i, j$ from layers $l, l + 1$ respectively.
Neural networks

\[ a^{(2)}_1 = g(w^{(1)}_{10} + w^{(1)}_{11} x_1 + w^{(1)}_{12} x_2 + w^{(1)}_{13} x_3) \]
\[ a^{(2)}_2 = g(w^{(1)}_{20} + w^{(1)}_{21} x_1 + w^{(1)}_{22} x_2 + w^{(1)}_{23} x_3) \]
\[ a^{(2)}_3 = g(w^{(1)}_{30} + w^{(1)}_{31} x_1 + w^{(1)}_{32} x_2 + w^{(1)}_{33} x_3) \]
\[ \text{Output} \rightarrow a^{(3)}_1 = g(w^{(2)}_{10} + w^{(2)}_{11} a^{(2)}_1 + w^{(2)}_{12} a^{(2)}_2 + w^{(2)}_{13} a^{(2)}_3) \]
Neural networks

Some useful names:

- **Feedforward neural network**: no cyclic connections between nodes from the same layer (previous example).
- **Multilayer perceptron (MLP)**: is a feedforward neural network with at least 3 layers.
- **Deep neural networks**: term referring to neural networks with more than one hidden layer.
Artificial neural networks architectures

Some examples of neural network popular architectures:

- **Recurrent neural networks**: neural networks where connections between nodes form a directed cycle.
  - built-in internal state memory
  - built-in notion of time ordering for a time sequence
Artificial neural networks architectures

- **Convolutional neural networks**: multilayer perceptron designed to require minimal preprocessing, *i.e.* space invariant architecture.
  - the hidden layers consist of convolutional layers, pooling layer, fully connected layers and normalization layers
  - great successful applications in image and video recognition.
Artificial neural networks architectures

- **Generative adversarial network**: unsupervised machine learning system of two neural networks contesting with each other.
  - one network generates candidates while the other discriminates.
Other popular examples:

- **Recursive neural networks**: a variation of recurrent neural network where pairs of layers or nodes are merged recursively.
  - successful applications on natural language processing.
  - some recent applications for model inference.
- **Long short-term memory**: another variation of recurrent neural networks composed by custom units cells:
  - LSTM cells have an input gate, an output gate and a forget gate.
  - powerful when making predictions based on time series data.
- **Boltzmann Machines**: is a generative stochastic recursive artificial neural network.
  - comes with energy-based model features and advantages.
From physics to ML
Introduction

Let's try to build a model:

- well suited for pdf estimation and pdf sampling
- built-in pdf normalization (close form expression)
- very flexible with a small number of parameters

We decided to look at energy models, specifically Boltzmann Machines.
Boltzmann machine

Graphical representation:
Boltzmann machine

Graphical representation:

[Hinton, Sejnowski ‘86]
Boltzmann machine

Graphical representation:

[Hinton, Sejnowski '86]
Boltzmann machine

Graphical representation:

[Hinton, Sejnowski ‘86]

Binary valued states \{0, 1\}
Boltzmann machine

Graphical representation:

[Hinton, Sejnowski ‘86]

Connection matrices

Binary valued states \{0, 1\}
Boltzmann machine

Graphical representation:

- Boltzmann machine (BM): $T$ and $Q \neq 0$.
- Restricted Boltzmann machine (RBM): $T = Q = 0$. 

[Hinton, Sejnowski ‘86]
Boltzmann machine

Energy based model:

The system energy for given state vectors \((v, h)\):

\[
E(v, h) = \frac{1}{2} v^t T v + \frac{1}{2} h^t Q h + v^t W h + B_h h + B_v v
\]

[Hinton, Sejnowski ‘86]
Boltzmann machine

Energy based model:

\[ E(v, h) = \frac{1}{2} v^t T v + \frac{1}{2} h^t Q h + v^t W h + B_h h + B_v v \]

View as statistical mechanical system.

State vectors \( \Rightarrow \) Connection matrices \( \Rightarrow \) Biases
Boltzmann machine

Energy based model: [Hinton, Sejnowski '86]

Starting from the system energy for given state vectors \((v, h)\):

\[
E(v, h) = \frac{1}{2} v^T T v + \frac{1}{2} h^T Q h + v^T W h + B_h h + B_v v
\]

The canonical partition function is defined as:

\[
Z = \sum_{h, v} e^{-E(v, h)}
\]

Probability the system is in specific state given by Boltzmann distribution:

\[
P(v, h) = \frac{e^{-E(v, h)}}{Z}
\]

with marginalization:

\[
P(v) = \frac{e^{-F(v)}}{Z} \quad \text{Free energy}
\]
Boltzmann machine

Learning:

Theoretically, general compute medium.
Via adjusting $W, T, Q, B_h, B_v$ able to learn the underlying probability distribution of a given dataset.

However: practically not feasible
For applications only RBMs have been considered.
Riemann-Theta Boltzmann machine

How to change the status quo? [Krefl, S.C., Haghighat, Kahlen '17]
Keep the inner sector couplings non-trivial, but the machine solvable?

$P(v) \equiv \text{multi-variate gaussian (too trivial)}$
**Riemann-Theta Boltzmann machine**

**How to change the status quo?**  
[Krefl, S.C., Haghighat, Kahlen ‘17]  
Keep the inner sector couplings non-trivial, but the machine solvable?

“Quantized”  
$\in \mathbb{Z}$

Continuous valued  
$\in \mathbb{R}$

**Something interesting happens**

Under mild constraints on connection matrices (positive definiteness, ...)

Closed form analytic solution still available!
Riemann-Theta Boltzmann machine

How to change the status quo? [Krefl, S.C., Haghighat, Kahlen ‘17]
Keep the inner sector couplings non-trivial, but the machine solvable?

"Quantized" ∈ \( \mathbb{Z} \)

Continuous valued ∈ \( \mathbb{R} \)

\[
P(v) \equiv \sqrt{\frac{\text{det} \, T}{(2\pi)^{N_v}}} e^{-\frac{1}{2} v^t T v - B_v v - \frac{1}{2} B_v T^{-1} B_v} \frac{\tilde{\theta}(B_h^t + v^t W | Q)}{\tilde{\theta}(B_h^t - B_v T^{-1} W | Q - W^t T^{-1} W)}
\]

Closed form analytic solution still available!
Riemann-Theta Boltzmann machine

**RTBM**

Novel very generic probability density:

\[
P(v) \equiv \sqrt{\frac{\det T}{(2\pi)^N_v}} e^{-\frac{1}{2}v^tTv - \frac{1}{2}B^t_vTv - \frac{1}{2}B^t_vT^{-1}B_v} \frac{\tilde{\theta}(B^t_h + v^tW|Q)}{\tilde{\theta}(B^t_h - B^t_vT^{-1}W|Q - W^tT^{-1}W)}
\]

Damping factor

\[
\tilde{\theta}(B^t_h - B^t_vT^{-1}W|Q - W^tT^{-1}W)
\]

Riemann-Theta function

The Riemann-Theta definition:

\[
\theta(z, \Omega) := \sum_{n \in \mathbb{Z}^{Nh}} e^{2\pi i (\frac{1}{2}n^t\Omega n + n^t z)}
\]

**Key properties:** Periodicity, modular invariance, solution to heat equation, etc.

**Note:** Gradients can be calculated analytically as well so gradient descent can be used for optimization.
We observe that $P(v)$ stays in the same distribution under affine transformations, i.e. rotation and translation

$$w = Av + b, \quad w \sim P_{A,b}(v),$$

if the linear transformation $A$ has full column rank.

$P_{A,b}(v)$ is the distribution $P(v)$ with parameters rotated as

$$T^{-1} \rightarrow AT^{-1}A^t, \quad B_v \rightarrow (A^+)^tB_v - Tb,$$

$$W \rightarrow (A^+)^tW, \quad B_h \rightarrow B_h - W^tb.$$

where $A^+$ is the left pseudo-inverse defined as

$$A^+ = (A^tA)^{-1}A^t.$$
In the next we show examples of RTBM for

- Probability determination
- Data classification
- Data regression
- Sampling
Riemann-Theta Boltzmann machine

RTBM $P(v)$ examples: [Kreft, S.C., Haghighat, Kahlen '17]

For different choices of parameters (with hidden sector in 1D or 2D).
Riemann-Theta Boltzmann machine

**Mixture model:**

**Expectation:**
As long as the density is well enough behaved at the boundaries it can be learned by an RTBM mixture model.

[Krefl, S.C., Haghighat, Kahlen ‘17]
Riemann-Theta Boltzmann machine

Examples: [Krefl, S.C., Haghighat, Kahlen ‘17]

Top $N_v = 1$, $N_h = 3, 2, 3$, button $N_v = 2$, $N_h = 1$ (2x RTBM), 2.
Riemann-Theta Boltzmann machine

Feature detector:

New:
Conditional expectations of hidden states after training

\[ E(h_i|v) = -\frac{1}{2\pi i} \frac{\nabla_i \tilde{\theta}(v^t W + B^t_h|Q)}{\tilde{\theta}(v^t W + B^t_h|Q)} \]

The detector is trained in probability mode and generates a feature vector.

[Krefl, S.C., Haghighat, Kahlen ‘17]
Similar to [Krizhevsky ‘09]
Jet classification:

Discriminating jets from single hadron particles and overlapping jets from pairs of collimated hadron particles.

Data (images of 32x32 pixels)

- 5000 images for training
- 2500 images for testing

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Test dataset precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression (LR)</td>
<td>77%</td>
</tr>
<tr>
<td>RTBM feature detector + LR</td>
<td>83%</td>
</tr>
</tbody>
</table>
**Theta Neural Network:**

**Idea:**
Use as activation function in a standard NN. The particular form of non-linearity is learned from data.

**Key point:**
smaller networks needed but Riemann-Theta evaluation is expensive.

**Example (1:3-3-2:1):**

\[
y(t) = 0.02t + 0.5 \sin(t + 0.1) + 0.75 \cos(0.25t - 0.3) + \mathcal{N}(0, 1)
\]
The probability for the visible sector can be expressed as:

\[ P(v) = \sum_{[h]} P(v|h)P(h) \]

where \( P(v|h) \) is a multivariate gaussian. The \( P(v) \) sampling can be performed easily by:

- sampling \( h \sim P(h) \) using the RT numerical evaluation \( \theta = \theta_n + \epsilon(R) \) with ellipsoid radius \( R \) so

\[ p = \frac{\epsilon(R)}{\theta_n + \epsilon(R)} \ll 1 \]

is the probability that a point is sampled outside the ellipsoid of radius \( R \), while

\[ \sum_{[h](R)} P(h) = \frac{\theta_n}{\theta_n + \epsilon(R)} \approx 1 \]

\( i.e. \) sum over the lattice points inside the ellipsoid.

- then sampling \( v \sim P(v|h) \)
Sampling examples

RTBM $P(v)$ sampling examples: [S.C. and Krefl ‘18]

Top $N_v = 1$, $N_h = 2, 3$ (2x RTBM), 3, button $N_v = 1$, $N_h = 3$. 
## Sampling distance estimators

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$\chi_{\text{RTBM}}/N_{\text{bins}}$</th>
<th>MSE$_{\text{RTBM}}$</th>
<th>MSE$_{\text{pdf}}$</th>
<th>KS distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma</td>
<td>0.02/50</td>
<td>$2 \cdot 10^{-5}$</td>
<td>$2.6 \cdot 10^{-5}$</td>
<td>0.01</td>
</tr>
<tr>
<td>Cauchy</td>
<td>0.12/50</td>
<td>$2.9 \cdot 10^{-4}$</td>
<td>$3.7 \cdot 10^{-4}$</td>
<td>0.02</td>
</tr>
<tr>
<td>Gaussian mixture</td>
<td>0.01/50</td>
<td>$6.7 \cdot 10^{-6}$</td>
<td>$1.4 \cdot 10^{-5}$</td>
<td>0.01</td>
</tr>
<tr>
<td>GOOG</td>
<td>0.10/50</td>
<td>$2.7 \cdot 10^{-4}$</td>
<td>$9.5 \cdot 10^{-3}$</td>
<td>0.02</td>
</tr>
<tr>
<td>XOM</td>
<td>0.09/50</td>
<td>$2.6 \cdot 10^{-4}$</td>
<td>$6.7 \cdot 10^{-3}$</td>
<td>0.02</td>
</tr>
</tbody>
</table>

**TABLE I**: Distance estimators for the sampling examples in figures [3] and [4]. Exact definitions for all distance estimators are given in section VII. The mean squared error (MSE) is taken between the sampling, the RTBM model and the underlying distribution (pdf). The Kolmogorov-Smirnov (KS) distance is shown in the last column of the table. For GOOG and XOM the empirical distribution is employed as underlying pdf.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Mean</th>
<th>2nd moment</th>
<th>3th moment</th>
<th>4th moment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gamma</td>
<td>7.43 (7.43) [7.49]</td>
<td>6.91 (6.89) [7.41]</td>
<td>10.03 (10.03) [13.79]</td>
<td>154 (153.23) [195.8]</td>
</tr>
<tr>
<td>Cauchy</td>
<td>-0.057 (-0.057) [-]</td>
<td>11.64 (11.64) [-]</td>
<td>-4.63 (-4.97) [-]</td>
<td>1749.8 (1753) [-]</td>
</tr>
<tr>
<td>Gaussian mixture</td>
<td>-1.48 (-1.48) [-1.31]</td>
<td>34.45 (34.45) [34.29]</td>
<td>134.35 (136.67) [131.78]</td>
<td>3558.7 (3571.8) [3569.1]</td>
</tr>
<tr>
<td>GOOG</td>
<td>0.06 (0.06) [0.08]</td>
<td>3.28 (3.23) [3.58]</td>
<td>1.52 (1.42) [6.04]</td>
<td>117 (108) [191]</td>
</tr>
<tr>
<td>XOM</td>
<td>0.02 (0.02) [0.03]</td>
<td>2.13 (2.15) [2.36]</td>
<td>-0.42 (-0.18) [1.44]</td>
<td>38.3 (40.2) [97.1]</td>
</tr>
</tbody>
</table>

**TABLE II**: Mean and central moments for the sampling data, the RTBM model (round brackets) and the underlying true distribution (square brackets). Note that the moments of the Cauchy distribution are either undefined or infinite. The given values correspond to the RTBM model approximation and its sampling, which are defined and finite, cf., [4]. For the GOOG and XOM distributions the true moments (square brackets) are evaluated from the underlying empirical distribution.
Sampling examples with affine transformation

**RTBM \( P(v) \) sampling with affine transformation:** [S.C. and Krefl ‘18]

For a rotation of \( \theta = \pi/4 \) and scaling of 2 \( (N_v = 2, N_h = 2) \).
Conclusion
In summary:

- ML is becoming very popular and strongly used in our field.
- Results are encouraging, several application opportunities.

For the future:

- New models based on physical systems.
- Try to extend the ML usage in physics.
Most popular public ML frameworks

For experimental HEP:

- TMVA: ROOT’s builtin machine learning package.

For ML applications:

- Keras: a Python deep learning library.
- Theano: a Python library for optimization.
- PyTorch: a DL framework for fast, flexible experimentation.
- Caffe: speed oriented deep learning framework.
- CNTK: Microsoft Cognitive Toolkit.
- Theta: the RTBM implementation library.

For ML and beyond:

- TensorFlow: library for numerical computation with data flow graphs.
- scikit-learn: general machine learning package.