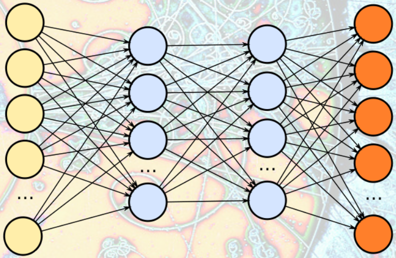
Big collaborations like **ATLAS**, **LZ** and **DUNE** spend a lot of computational resources on

- event generation,
- event reconstruction, and
- analysis,

with the goal of obtaining the best possible results. A lot of these techniques rely on machine learning algorithms.

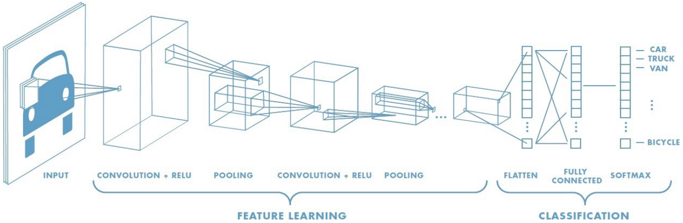


How can we be sure that these algorithms are performing optimally and efficiently?

Mutual Information and its Application to Machine Learning in HEP$^{1-6}$

Nicholas Carrara[†]
Ph.D. Candidate

University at Albany



† N. Carrara (Thesis) "The Foundations of Inference and its Application to Fundamental Physics" (2021)

1 N. Carrara and K. Vanslette, "The Design of Global Correlation Quantifiers and Continuous Notions of Statistical Sufficiency", *Entropy* 2020, 22(3), 357

2 N. Carrara and J. Ernst, "On the Upper Limit of Separability", arXiv:1708.09449

3 N. Carrara and J. Ernst, "On the Estimation of Mutual Information", arXiv:1910.00365

4 The LUX Collaboration, "Combining Machine Learning and Profile Likelihood Methods on the LUX Experiment", *In preparation*

5 N. Carrara and J. Ernst, "The Upper-Limit of Separability", *In preparation*

6 N. Carrara and J. Ernst, "The MIST Algorithm", *In preparation*

How is mutual information useful for HEP?

Mutual information is an information theoretic quantity (a relative entropy) which measures the *amount* of correlations[1] between two sets of variables X and Y,

$$I[X;Y] = \int dxdy\, p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

How is mutual information useful for HEP?

Mutual information is an information theoretic quantity (a relative entropy) which measures the *amount* of correlations[1] between two sets of variables X and Y,

$$I[X;Y] = \int dxdy\, p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

It defines an **upper-limit of separability**[2] which allows one to know:

How is mutual information useful for HEP?

Mutual information is an information theoretic quantity (a relative entropy) which measures the *amount* of correlations[1] between two sets of variables X and Y,

$$I[X;Y] = \int dxdy\, p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

It defines an **upper-limit of separability**[2] which allows one to know:

1. when to stop training any ML algorithm,

How is mutual information useful for HEP?

Mutual information is an information theoretic quantity (a relative entropy) which measures the *amount* of correlations[1] between two sets of variables X and Y,

$$I[X;Y] = \int dxdy\, p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

It defines an **upper-limit of separability**[2] which allows one to know:

1. when to stop training any ML algorithm,
2. when variables are redundant or useless[3],

How is mutual information useful for HEP?

Mutual information is an information theoretic quantity (a relative entropy) which measures the *amount* of correlations[1] between two sets of variables X and Y,

$$I[X;Y] = \int dxdy\, p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

It defines an **upper-limit of separability**[2] which allows one to know:

1. when to stop training any ML algorithm,
2. when variables are redundant or useless[3],
3. which variables from a given set are the best to use[4,5],

How is mutual information useful for HEP?

Mutual information is an information theoretic quantity (a relative entropy) which measures the *amount* of correlations[1] between two sets of variables X and Y,

$$I[X;Y] = \int dxdy\, p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

It defines an **upper-limit of separability**[2] which allows one to know:

1. when to stop training any ML algorithm,
2. when variables are redundant or useless[3],
3. which variables from a given set are the best to use[4,5],
4. quantifies systematic uncertainties in models[1],

How is mutual information useful for HEP?

Mutual information is an information theoretic quantity (a relative entropy) which measures the *amount* of correlations[1] between two sets of variables X and Y,

$$I[X;Y] = \int dxdy\, p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

It defines an **upper-limit of separability**[2] which allows one to know:

1. when to stop training any ML algorithm,
2. when variables are redundant or useless[3],
3. which variables from a given set are the best to use[4,5],
4. quantifies systematic uncertainties in models[1],
5. and more...

1 N. Carrara and K. Vanslette, "The Design of Global Correlation Quantifiers and Continuous Notions of Statistical Sufficiency", *Entropy* 2020, 22(3), 357

2 N. Carrara and J. Ernst, "On the Upper Limit of Separability", arXiv:1708.09449

3 N. Carrara and J. Ernst, "On the Estimation of Mutual Information", arXiv:1910.00365

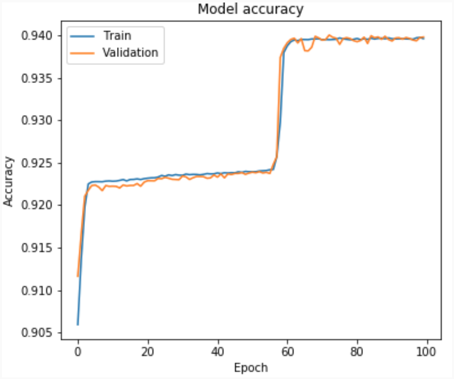
5 N. Carrara and J. Ernst, "The Upper-Limit of Separability", *In preparation*

6 N. Carrara and J. Ernst, "The MIST Algorithm", *In preparation*

The Upper-Limit of Separability

One of the prevailing challenges in generic machine learning tasks is to know when to stop training. Standard techniques only provide heuristics for when to stop.

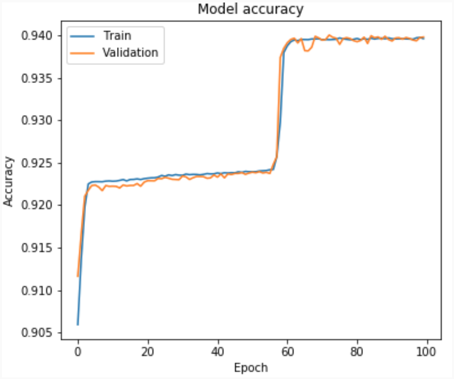
- This model is clearly stuck in a local minimum between epochs **10-60**.
- How can we be sure that it's not in a local minimum at epoch 100?



The Upper-Limit of Separability

One of the prevailing challenges in generic machine learning tasks is to know when to stop training. Standard techniques only provide heuristics for when to stop.

- This model is clearly stuck in a local minimum between epochs **10-60**.
- How can we be sure that it's not in a local minimum at epoch 100?



Our method, developed in "On the Upper Limit of Separability",

1. provides an **absolute target** (the mutual information) for when to stop training any algorithm, and
2. is quickly computable from data[7,8].

7 A. Kraskov, H. Stögbauer, and P. Grassberger, *Estimating Mutual Information*, Phys. Rev. E 69, 2004
8 G. ver Steeg, *The Non-parametric Entropy Estimation Toolbox*, `https://github.com/gregversteeg/NPEET`

Redundancy in HEP searches

The upper-limit is able to recognize the existence of **redundant** information.

Redundancy in HEP searches

The upper-limit is able to recognize the existence of **redundant** information.
In HEP[9] searches/event reconstructions you have

- some **low-level kinematic** variables X_{low},
  (**jet momenta, pseudo-rapidity, scintillation/ionization counts,etc.**)

Redundancy in HEP searches

The upper-limit is able to recognize the existence of **redundant** information.
In HEP[9] searches/event reconstructions you have

- some **low-level kinematic** variables X_{low},
  (**jet momenta, pseudo-rapidity, scintillation/ionization counts,etc.**)
- as well as **high-level** variables,

$$X_{\text{high}} = f(X_{\text{low}}) \tag{2}$$

  which can be outputs of,

  1. a ML algorithm[10],
  2. an event reconstruction algorithm,
     (**invariant masses, pulse shape variables, etc.**)
  3. etc.

Redundancy in HEP searches

The upper-limit is able to recognize the existence of **redundant** information. In HEP[9] searches/event reconstructions you have

- some **low-level kinematic** variables X_{low},

  (**jet momenta, pseudo-rapidity, scintillation/ionization counts,etc.**)

- as well as **high-level** variables,

$$X_{\mathrm{high}} = f(X_{\mathrm{low}}) \qquad (2)$$

  which can be outputs of,

  1. a ML algorithm[10],
  2. an event reconstruction algorithm,

     (**invariant masses, pulse shape variables, etc.**)
  3. etc.

The upper-limit guarantees that,

$$I[(X_{\mathrm{low}}, X_{\mathrm{high}}); \theta] = I[X_{\mathrm{low}}; \theta]. \qquad (3)$$

Redundancy in HEP searches

The upper-limit is able to recognize the existence of **redundant** information.

In HEP[9] searches/event reconstructions you have

- some **low-level kinematic** variables X_{low},

  (**jet momenta, pseudo-rapidity, scintillation/ionization counts,etc.**)
- as well as **high-level** variables,

$$X_{\text{high}} = f(X_{\text{low}}) \qquad (2)$$

which can be outputs of,

1. a ML algorithm[10],
2. an event reconstruction algorithm,

   (**invariant masses, pulse shape variables, etc.**)
3. etc.

The upper-limit guarantees that,

$$I[(X_{\text{low}}, X_{\text{high}}); \theta] = I[X_{\text{low}}; \theta]. \qquad (3)$$

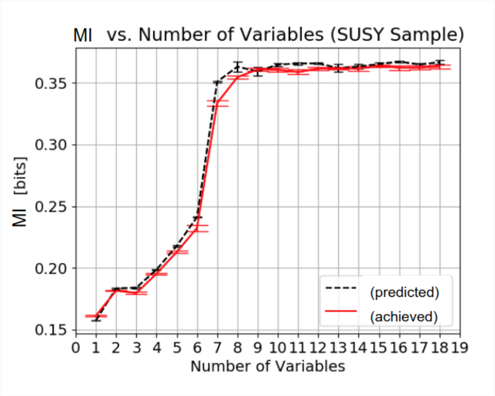


Figure 1: MI vs. number of variables for the input variables X (predicted) and the output of the trained model $f(X)$ (acheived).

9 The SUSY Dataset, UCI Machine learning repository, `https://archive.ics.uci.edu/ml/datasets/SUSY`

10 Komiski et al., "Energy Flow Networks: Deep Sets for Particle Jets", `https://arxiv.org/abs/1810.05165`

Kaggle Higgs ML Challenge

Thousands of teams competed in the challenge[11], using all sorts of techniques,

- deep neural networks,
- boosted decision trees,
- random forests,
- ⋮

with the highest placed entries clustering around a top score of (AMS = 3.8).

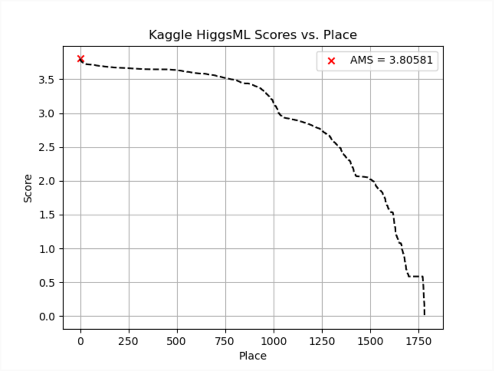


Figure 2: AMS (Average mean-significance) scores for entries in the Kaggle HiggsML challenge[11]. The top entry (Gabor Melis) is marked with an (x).

Main difficulty: The data set contains a large number of variables (**30**), most of which contain **redundant/useless** information.

11 Higgs Boson Machine Learning Challenge, Kaggle, https://www.kaggle.com/c/higgs-boson

Feature Selection

Using a **reinforcement learning** algorithm we developed called **MIST** (the mutual information search tree)

Feature Selection

> Using a **reinforcement learning** algorithm we developed called **MIST** (the mutual information search tree)

we achieved the
following,

Feature Selection

Using a **reinforcement learning** algorithm we developed called **MIST** (the mutual information search tree)

we achieved the following,

- algorithm only took **20 minutes** to run,

Feature Selection

Using a **reinforcement learning** algorithm we developed called **MIST** (the mutual information search tree)

we achieved the following,

- algorithm only took **20 minutes** to run,
- it requires **no training of** any models on subspaces,

Feature Selection

Using a **reinforcement learning** algorithm we developed called **MIST** (the mutual information search tree)

we achieved the following,

- algorithm only took **20 minutes** to run,
- it requires **no training of** any models on subspaces,
- it extracted **9** features from the set of **30** which contain **all the relevant information**, and

Feature Selection

Using a **reinforcement learning** algorithm we developed called **MIST** (the mutual information search tree)

we achieved the following,

- algorithm only took **20 minutes** to run,
- it requires **no training of** any models on subspaces,
- it extracted **9** features from the set of **30** which contain **all the relevant information**, and
- computed the **upper-limit** for the entire data set.

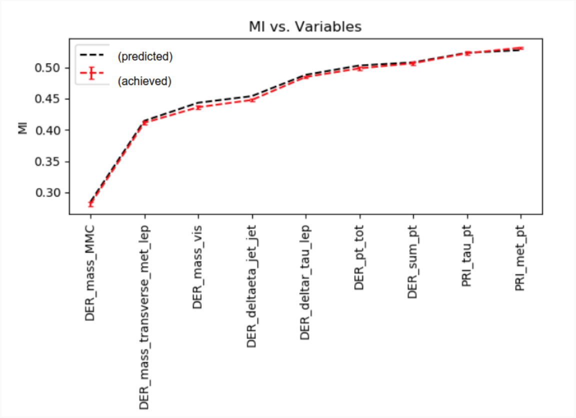


Figure 3: MI vs. number of variables for the winning input variables X (predicted) and the output of a trained model[12] $f(X)$ (acheived).

12 Strong, Giles, On the impact of selected modern deep-learning techniques to the performance and celerity of classification models in an experimental high-energy physics use case, Mach. Learn.: Sci. Technol (2020)

Combining ML and PLR on LUX

We combined an ML algorithm with the standard PLR approach on LUX.

Combining ML and PLR on LUX

We combined an ML algorithm with the standard PLR approach on LUX.

This resulted in,

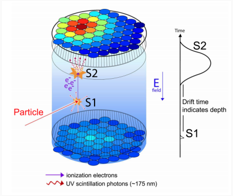
1. an **order of magnitude lower** computational time – ($\sim 600 \rightarrow 60$ min.), and

Combining ML and PLR on LUX

We combined an ML algorithm with the standard PLR approach on LUX.

This resulted in,

1. an **order of magnitude lower** computational time – ($\sim 600 \rightarrow 60$ min.), and
2. the option to add additional variables without increasing computational complexity (e.g. pulse shape)[4].



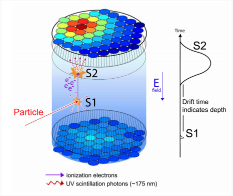
S2
S1
Particle
E_{field}
Time
S2
Drift time indicates depth
S1
ionization electrons
UV scintillation photons (~175 nm)

Combining ML and PLR on LUX

We combined an ML algorithm with the standard PLR approach on LUX.

This resulted in,

1. an **order of magnitude lower** computational time – ($\sim 600 \rightarrow 60$ min.), and
2. the option to add additional variables without increasing computational complexity (e.g. pulse shape)[4].


S2
S1
Particle
E field
Time
S2
Drift time indicates depth
S1
ionization electrons
UV scintillation photons (~175 nm)

Using the ML output $f(X)$ in-place of the original variables X we achieved an identical result to the Run03 re-analysis[4].

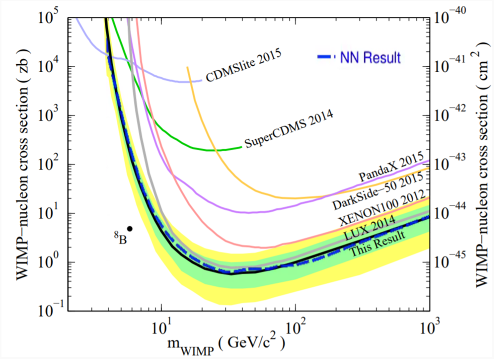


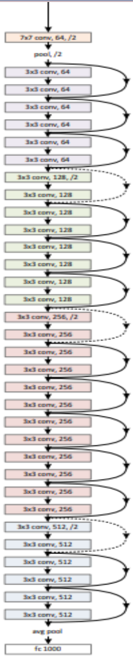
Figure 4: PLR limits on the neural network output for Run03.

4 The LUX Collaboration, "Combining Machine Learning and Profile Likelihood Methods on the LUX Experiment", *In preparation*

Summary

Topics of the rest of the talk:

1. **The Upper-Limit of Separability**: Demonstrating MI as an **absolute metric** for training discriminators
   - Basic examples
   - Practical HEP example - mock SUSY data set
2. **The Mutual Information Search Tree**: The upper-limit as a variable/feature selection criteria in **reinforcement learning**
   - Pitfalls of the MI estimation scheme
   - Monte Carlo Tree Search
   - MIST Demo!
3. **Combining ML with PLR on LUX**: Reducing computational overhead by constructing **sufficient statistics**
   - The LUX experiment
   - The Profile Likelihood Ratio Method
   - Using ML and the MI to supplement the PLR



The Upper-Limit of Separability

Mutual information as a hard-target for training discriminators

Mutual Information

Mutual information has been utilized throughout many signal processing and machine learning paradigms,

- **Rate-Distortion Theory** - *minimization* problem (Shannon, 1948[13]).
- **InfoMax Principle** - *maximization* problem (Linsker, 1988[14])
- **The Information Bottleneck Method** - *min-max* problem (extension of rate-distortion theory) (Tishby et al., 2000[15])
- **InfoGAN** - *maximization* problem, using MI to help disentangle representations (Chen et al., 2016[16]).
- **Inception Score** - *metric* used for evaluating generative models (Salimans et al., 2016[17])
- **Separability Principle** - *optimization* problem in which the MI is a metric of performance (Carrara, Ernst, 2017[2]).
- **n-partite Sufficiency** - continuous *measures* of statistical sufficiency (Carrara, Vanslette, 2020[1]).

13 Shannon, Claude Elwood (July 1948). "A Mathematical Theory of Communication" (PDF). Bell System Technical Journal. 27 (3): 379–423

14 Linsker R (1988). "Self-organization in a perceptual network". IEEE Computer. 21 (3): 105–17

15 Tishby et al. (2000). `https://arxiv.org/abs/physics/0004057`

16 Chen et al. (2016). `https://arxiv.org/abs/1606.03657`

17 Salimans et al. "Improved techniques for training GANs", `https://arxiv.org/pdf/1606.03498.pdf`

"Global Correlation Quantifiers"

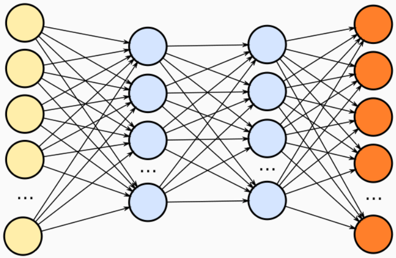
In [1,2], we showed that by taking any features (X) through some ML algorithm ($f(X)$) that

$$I[X;\theta] \geq I[f(X);\theta] \tag{4}$$

“Global Correlation Quantifiers”

In [1,2], we showed that by taking any features (X) through some ML algorithm ($f(X)$) that

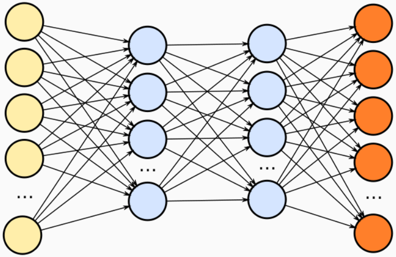
$$I[X;\theta] \geq I[f(X);\theta] \qquad (4)$$



“Global Correlation Quantifiers”

In [1,2], we showed that by taking any features (X) through some ML algorithm ($f(X)$) that

$$I[X;\theta] \geq I[f(X);\theta] \tag{4}$$



Then, by defining the **sufficiency**,

$$\mathrm{suff}_\Theta[f(\mathrm{X})] = \frac{I[f(\mathrm{X});\Theta]}{I[\mathrm{X};\Theta]}, \tag{5}$$

one has a **measure** of the **systematic uncertainties** in the model $f(\mathrm{X})$.

1 N. Carrara and K. Vanslette, “The Design of Global Correlation Quantifiers and Continuous Notions of Statistical Sufficiency”, *Entropy* 2020, 22(3), 357

2 N. Carrara and J. Ernst, “On the Upper Limit of Separability”, arXiv:1708.09449

“On the Upper-Limit of Separability”

In “The Upper-Limit of Seperability”[2] we demonstrated the following properties of MI,

“On the Upper-Limit of Separability”

In “The Upper-Limit of Seperability”[2] we demonstrated the following properties of MI,

- The MI is an **upper-limit** for any ML algorithm (eq. 4),

“On the Upper-Limit of Separability”

In “The Upper-Limit of Seperability”[2] we demonstrated the following properties of MI,

- The MI is an **upper-limit** for any ML algorithm (eq. 4),
- the upper-limit is insensitive to **redundant information** (eq. 3), and

“On the Upper-Limit of Separability”

In “The Upper-Limit of Seperability”[2] we demonstrated the following properties of MI,

- The MI is an **upper-limit** for any ML algorithm (eq. 4),
- the upper-limit is insensitive to **redundant information** (eq. 3), and
- it provides a **measure of separation** between two probability distributions.

“On the Upper-Limit of Separability”

In “The Upper-Limit of Seperability”[2] we demonstrated the following properties of MI,

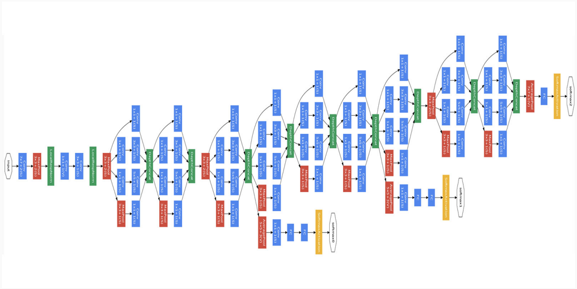
- The MI is an **upper-limit** for any ML algorithm (eq. 4),
- the upper-limit is insensitive to **redundant information** (eq. 3), and
- it provides a **measure of separation** between two probability distributions.

By **optimizing** a network so that

$$I[X;\theta] = I[f(X);\theta] \quad (6)$$

Then it's **guaranteed** that

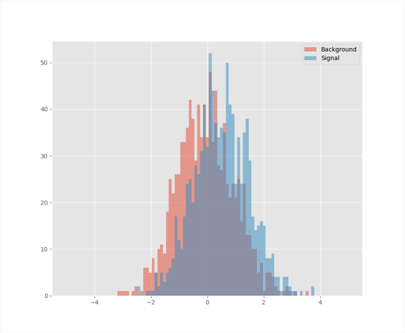
$$p(\theta|x) = p(\theta|f(x)) \quad (7)$$



N.B. When (6) is satisfied, the type I (false positives) and type II (false negatives) errors associated with θ will be identical for X and $f(X)$!

The Jensen-Shannon Divergence

In binary classification tasks MI is also equivalent to the *Jensen-Shannon divergence* (JSD)[18,19], which measures the *separation* between two distributions.



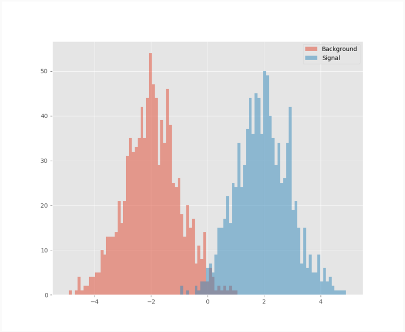


Figure 5: Examples of two distributions which are (left) mostly indistinguishable (**low separation/low MI value**) and (right) highly distinguishable (**large separation/high MI value**)

18 Chen et al., "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets", `https://arxiv.org/abs/1606.03657`

19 Salimans et al., "Improved techniques for training GANs", `https://arxiv.org/abs/1606.03498`

Tests of Separation

Is the upper-limit a measure of **separation**?

Consider a five-dimensional spherical Gaussian with $\sigma_i = 1$.

- Generate signal (s) and background (b) distributions with varying $\Delta\mu = |\mu_s - \mu_b|$.
- Compute the MI before and after training a neural network.

The result shows that MI is a function of the separation $\Delta\mu$.

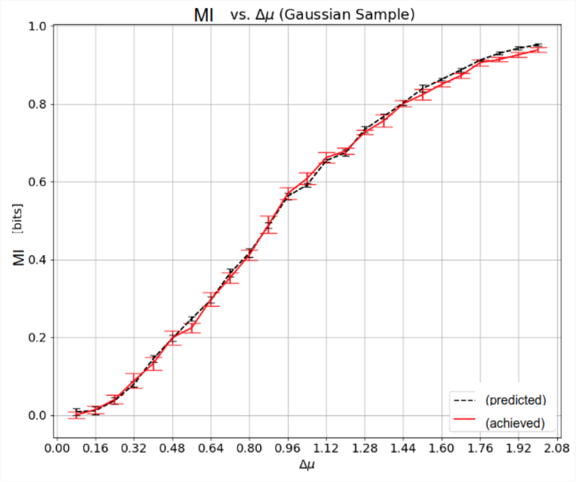


Figure 6: MI of five Gaussian variables vs. separation $\Delta\mu = |\mu_s - \mu_b|$ before (predicted) and after (achieved) training a neural network.

Practical HEP example: SUSY search

This practical example consists of a mock SUSY search[20,21] which

- contains **8** low-level (primitive) variables, (**jet momenta, azimuthal angle, pseudo-rapidity**)
- contains **10** high-level (derived) variables[22]. (**invariant masses, razor and super-razor quantities**)

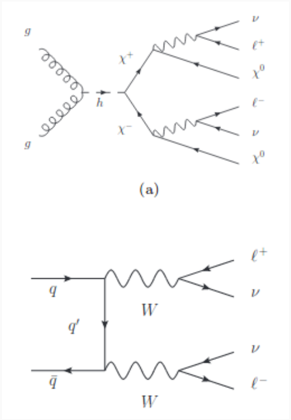


Figure 7: Signal $((\chi^+\chi^-) \rightarrow (\chi^0 W \chi^0 W) \rightarrow (\chi^0\chi^0)(\ell^+\ell^-\nu\nu))$ and background $((WW) \rightarrow (\ell^+\ell^-\nu\nu))$ processes for a SUSY search.

20 Baldi et. al., Searching for Exotic Particles in High-Energy Physics with Deep Learning, Nature Communications volume 5, 4308 (2014)

21 SUSY Data set, `https://archive.ics.uci.edu/ml/datasets/SUSY`

22 Buckley et. al., Super-Razor and Searches for Sleptons and Charginos at the LHC, Phys. Rev. D 89, 055020 (2014)

SUSY dataset results

The MI predicts that the **8** low-level variables contain all relevant information. After training our own neural network we find,

- Each model trained on a variable subset approaches the upper-limit (achieved).
- The low-level (primitive) variables achieve the same separation as the high-level (derived) variables.

MI	Low only	High only	Both
(predicted)	0.36(2)	0.36(2)	0.37(2)
(achieved)	0.35(2)	0.35(2)	0.36(2)

Table 1: MI values on the input variables (predicted) and for our network (achieved) outputs.

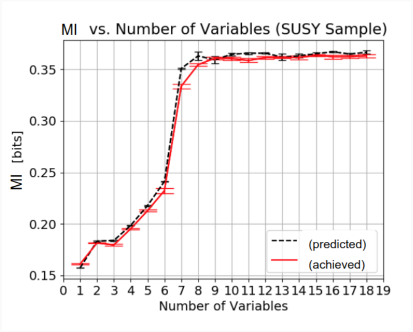


Figure 8: MI vs. variables for the low-level (first 8) and subsequent high-level (last 10) variables in the SUSY dataset.

SUSY dataset results: AUC comparisons

To compare our neural network model to the one from the paper,

- we compute the **area under curve** (AUC) of our model output.
- compare our AUC to the ones from the paper (BSW)[20].

MI	Low only	High only	Both
(predicted)	0.36(2)	0.36(2)	0.37(2)
(achieved)	0.35(2)	0.35(2)	0.36(2)

AUC	Low only	High only	Both
(this work)	0.87	0.87	0.88
(BSW, shallow)	0.86	0.86	0.88
(BSW, deep)	0.88	0.87	0.88

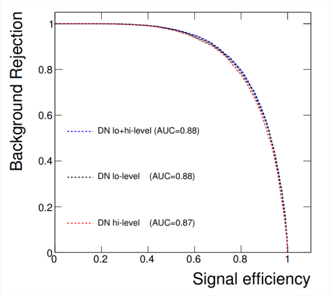


Figure 9: ROC curves from the BSW paper for deep neural network outputs trained on the SUSY dataset[20].

20 Baldi et. al. (BSW), Searching for Exotic Particles in High-Energy Physics with Deep Learning, Nature Communications volume 5, 4308 (2014)

The Mutual Information Search Tree

Reinforcement learning for feature selection

Pitfalls of MI estimation

The widely used MI estimator developed by Kraskov et al. (KSG)[22] has some shortcomings.

Pitfalls of MI estimation

The widely used MI estimator developed by Kraskov et al. (KSG)[22] has some shortcomings.

- Accuracy in large dimensions depends on the size of the sample distribution.

Pitfalls of MI estimation

The widely used MI estimator developed by Kraskov et al. (KSG)[22] has some shortcomings.

- Accuracy in large dimensions depends on the size of the sample distribution.
- **Useless variables drastically affect the estimation.**

Pitfalls of MI estimation

The widely used MI estimator developed by Kraskov et al. (KSG)[22] has some shortcomings.

- Accuracy in large dimensions depends on the size of the sample distribution.
- **Useless variables drastically affect the estimation.**

To demonstrate this, consider the SUSY search and

1. take each high level variable (derived) and **shuffle** them within signal/background to make them **useless**, and
2. repeat same procedure as in the previous slide

Pitfalls of MI estimation

The widely used MI estimator developed by Kraskov et al. (KSG)[22] has some shortcomings.

- Accuracy in large dimensions depends on the size of the sample distribution.
- **Useless variables drastically affect the estimation.**

To demonstrate this, consider the SUSY search and

1. take each high level variable (derived) and **shuffle** them within signal/background to make them **useless**, and
2. repeat same procedure as in the previous slide

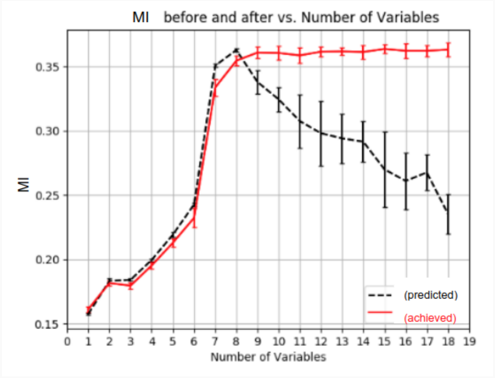


Figure 10: MI vs. variables for the SUSY data set where the high level input variables are shuffled within signal/background[23].

22 Kraskov, A., Stögbauer, H. and Grassberger, P., *Estimating Mutual Information*, `https://arxiv.org/abs/cond-mat/0305641`
23 N. Carrara and J. Ernst, On the estimation of mutual information, MaxEnt 2019, `https://arxiv.org/abs/1910.00365`

How do we “see” through useless variables?

Given no knowledge of the shuffling of the high-level variables, how could we know to ignore them?

How do we "see" through useless variables?

Given no knowledge of the shuffling of the high-level variables, how could we know to ignore them?

1. One potential solution would be to compute the MI on **all 2^d subspaces** for a d-dimensional variable space.

How do we “see” through useless variables?

Given no knowledge of the shuffling of the high-level variables, how could we know to ignore them?

1. One potential solution would be to compute the MI on **all** 2^d **subspaces** for a d-dimensional variable space.
2. The SUSY example would require $2^{18} = 262,144$ MI calculations, which with an average of $\approx 60s$ each would take ≈ 6 months to complete!

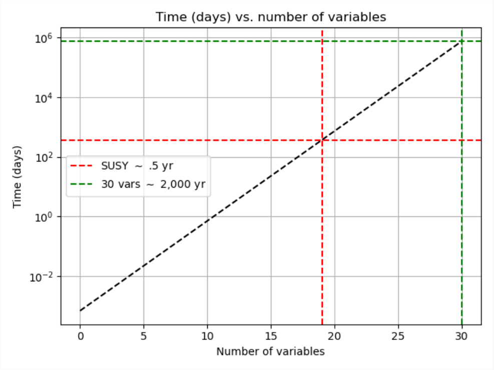


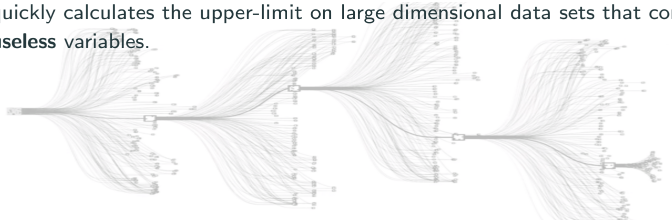
Figure 11: Computation time to calculate MI on every subset of a d-dimensional space, assuming one subset takes $\approx 60s$.

The Mutual Information Search Tree

MIST

We developed a machine learning algorithm (reinforcement learning) called the mutual information search tree (MIST)[5,6] which,

1. Searches the subspaces of variable space to find a subset which contains the **maximum amount of information**,
2. does not require the training of **any neural networks**, and
3. quickly calculates the upper-limit on large dimensional data sets that contain **useless** variables.



5 N. Carrara and J. Ernst, "The Upper-Limit of Separability", *In preparation*
6 N. Carrara and J. Ernst, "The MIST Algorithm", *In preparation*

Example Tree

The following is an example of a playout for a tree consisting of three variables,

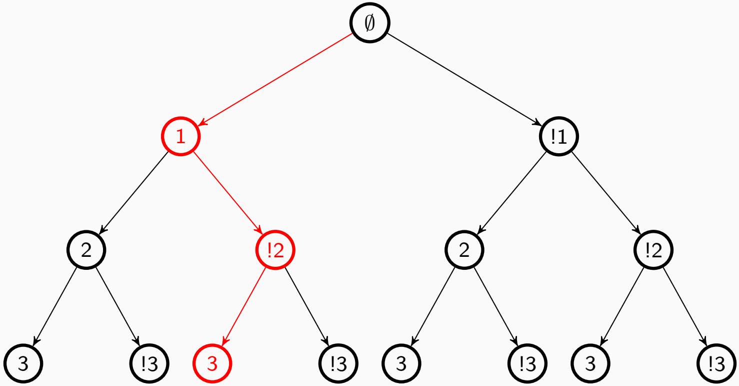


Figure 12: Simple MIST search tree for three variables. Each node propagated from the left corresponds to a variable that is kept, while each node propagated from the right is a variable that is ignored. The path highlighted in red corresponds to a playout in which the variables (1) and (3) are kept, but variable (2) is ignored so that $X_p = \{1, 3\}$.

MIST Demo

Let's take a look at how the MIST algorithm works!

MIST Demo

Let's take a look at how the MIST algorithm works!

To recap,

MIST Demo

Let's take a look at how the MIST algorithm works!

To recap,

- MIST easily finds subspaces of variable space in the SUSY example which contain the **most useful information**.

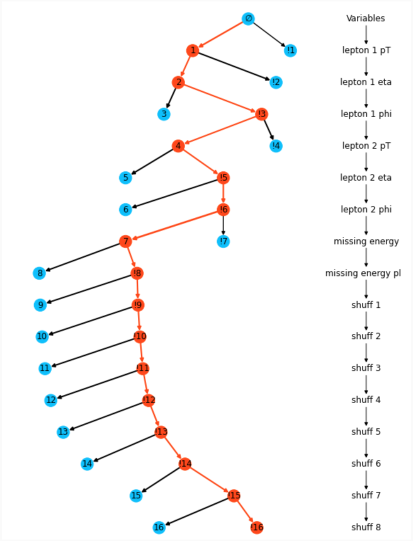
MIST Demo

Let's take a look at how the MIST algorithm works!

To recap,

- MIST easily finds subspaces of variable space in the SUSY example which contain the **most useful information**.
- The algorithm **never** has to train any networks – for each playout we only need to compute the MI.

The example to the right shows the algorithm selecting variables $\{1, 2, 4, 7\}$ and throwing away everything else.



The Higgs data set

The Higgs data set contains,

- **17** low-level (primitive) features,
- **13** high-level (derived) features

The top contestants clustered around a value of AMS ~ 3.8.

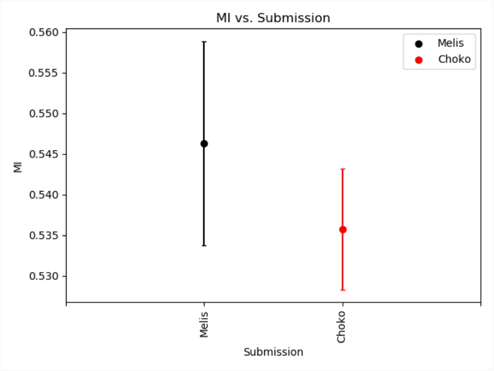


Figure 13: MI scores on the output of the first place (Melis) and third place (Choko) submissions to the Kaggle HiggsML challenge.

MIST Results on the Higgs

Using MIST on the Higgs data set
we find,

MIST Results on the Higgs

Using MIST on the Higgs data set we find,

- algorithm takes about **20 minutes** to run,

MIST Results on the Higgs

Using MIST on the Higgs data set we find,

- algorithm takes about **20 minutes** to run,
- extracts **9** features from the set of **30** which captures the upper-limit of **.535** $\pm$ **.015**.

MIST Results on the Higgs

Using MIST on the Higgs data set we find,

- algorithm takes about **20 minutes** to run,
- extracts **9** features from the set of **30** which captures the upper-limit of **.535** $\pm$ **.015**.

To compare with the top contestants, we use a trained model developed by Giles Strong[12,24] to evaluate the performance of the **9** winning variables.

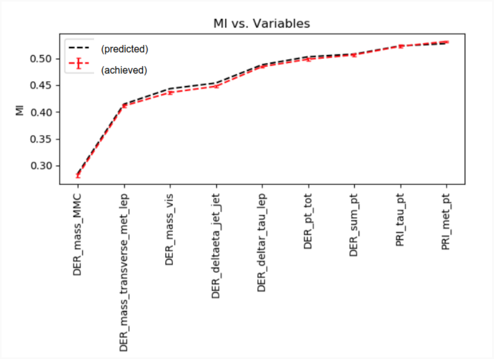


Figure 14: Running totals of MI of each of nine variables in the winning MIST set. The net output (achieved) corresponds to the model developed by Giles Strong[12,24].

12 Giles Strong, *On the impact of selected modern deep-learning techniques to the performance and celerity of classification models in an experimental high-energy physics use case*, `https://arxiv.org/pdf/2002.01427.pdf`

24 Giles Strong, Code for the HiggsML challenge, `https://github.com/GilesStrong/HiggsML_Lumin`

Higgs results cont.

In our study we found,

- Error bars on the MIST prediction tend to be larger since the MI estimate is done over a **9** dimensional space[25].

Higgs results cont.

In our study we found,

- Error bars on the MIST prediction tend to be larger since the MI estimate is done over a **9** dimensional space[25].
- Prediction on the MIST variables captures Giles model on all variables **(All)** and on the MIST subset **(MIST)**.
- Both **(Choko)** and **(Melis)** used additional variables, so an apples to apples comparison is not possible.

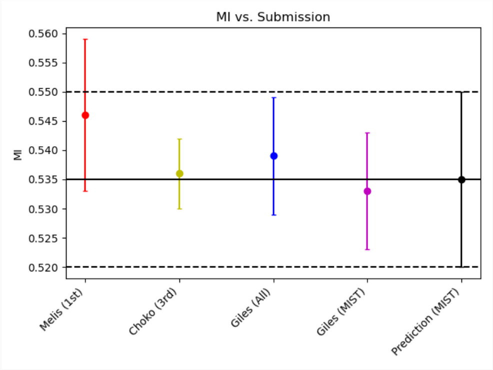


Figure 15: MI values corresponding to the 1st place (Melis) and 3rd place (Choko) submissions, along with the prediction from MIST, and the output of Giles Strong's models on (All) variables and the **9** predicted from (MIST).

25 Holmes and Nemenman, "Estimation of mutual information for real-valued data with error bars and controlled bias", https://journals.aps.org/pre/abstract/10.1103/PhysRevE.100.022404

Higgs results cont.

Comparing AMS scores of Giles output on All and MIST variables,

Submission	AMS	MI
Prediction	–	$.535 \pm .015$
Giles (All)	$3.8 \pm .05$	$.539 \pm .01$
Giles (MIST)	$3.7 \pm .1$	$.533 \pm .01$
Melis (1st)	3.8	$.545 \pm .015$
Choko (3rd)	3.77	$.536 \pm .005$

Table 2: AMS scores and MI values for the MIST prediction ($.535 \pm .015$) and the outputs of Giles, Melis and Choko models.

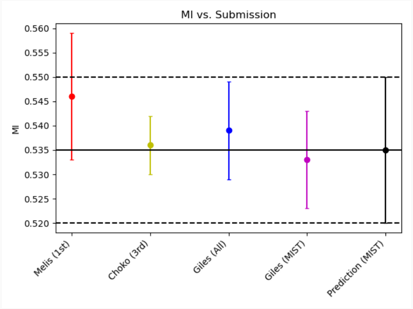


Figure 16: MI values corresponding to the 1st place (Melis) and 3rd place (Choko) submissions, along with the prediction from MIST, and the output of Giles Strong's models on (All) variables and the **9** predicted from (MIST).

Higgs results cont.

Comparing to Giles' study[24],

- we rank the **importance** of each of the **9** winning variables by their individual upper-limit scores.

Higgs results cont.

Comparing to Giles' study[24],

- we rank the **importance** of each of the **9** winning variables by their individual upper-limit scores.
- The ranking matches that determined by traditional **feature permutation methods**.

Higgs results cont.

Comparing to Giles' study[24],

- we rank the **importance** of each of the **9** winning variables by their individual upper-limit scores.
- The ranking matches that determined by traditional **feature permutation methods**.

The benefit of MIST is that,

- Ranking of **importance** is done at the level of entire **subsets** rather than individual variables,
- which means that the ranking takes into account **all of the correlations**.

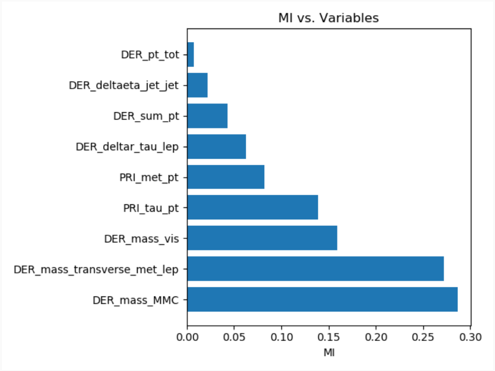


Figure 17: Individual upper-limit scores for each of the **9** variables in the winning set.

Combining Machine Learning and Profile Likelihood Methods on LUX

The Liquid Underground Xenon (LUX) experiment

LUX is a dual phase (gas/liquid) Xenon TPC (time projection chamber) direct detection experiment searching for dark matter.

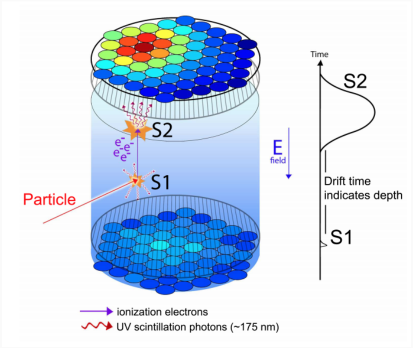


Figure 18: Diagram of the dual phase Xenon TPC in the LUX experiment which consists of two arrays of PMTs.

The Liquid Underground Xenon (LUX) experiment

LUX is a dual phase (gas/liquid) Xenon TPC (time projection chamber) direct detection experiment searching for dark matter.

Events of interest consist of the following kinematical steps:

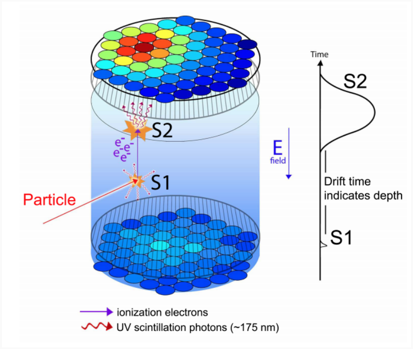


Figure 18: Diagram of the dual phase Xenon TPC in the LUX experiment which consists of two arrays of PMTs.

The Liquid Underground Xenon (LUX) experiment

LUX is a dual phase (gas/liquid) Xenon TPC (time projection chamber) direct detection experiment searching for dark matter.

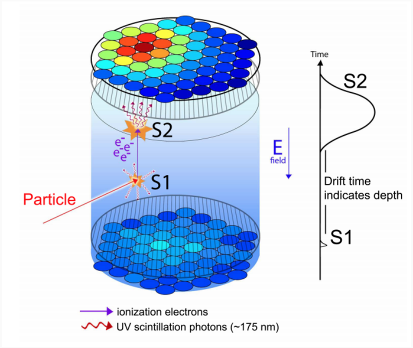

Time
S2
S2
E field
S1
Particle
Drift time indicates depth
S1
ionization electrons
UV scintillation photons (~175 nm)

Figure 18: Diagram of the dual phase Xenon TPC in the LUX experiment which consists of two arrays of PMTs.

Events of interest consist of the following kinematical steps:

1. Incident particles interact with the liquid Xenon (LXe) causing either a **nuclear or electron recoil event**. (Results are insensitive to the underlying mediator particle).

The Liquid Underground Xenon (LUX) experiment

LUX is a dual phase (gas/liquid) Xenon TPC (time projection chamber) direct detection experiment searching for dark matter.

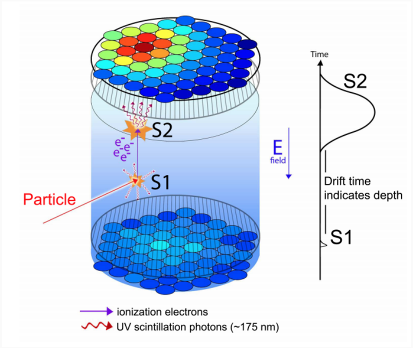


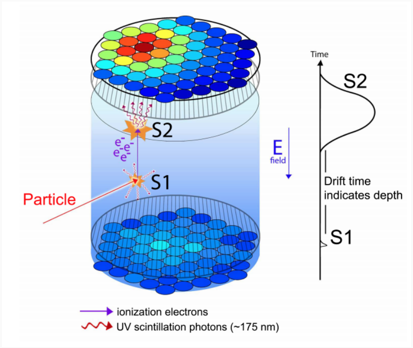
Figure 18: Diagram of the dual phase Xenon TPC in the LUX experiment which consists of two arrays of PMTs.

Events of interest consist of the following kinematical steps:

1. Incident particles interact with the liquid Xenon (LXe) causing either a **nuclear or electron recoil event**. (Results are insensitive to the underlying mediator particle).
2. The recoil event releases photons **(S1)** and ionization electrons.

The Liquid Underground Xenon (LUX) experiment

LUX is a dual phase (gas/liquid) Xenon TPC (time projection chamber) direct detection experiment searching for dark matter.



Time
S2
S2
E field
Drift time indicates depth
S1
S1
Particle
ionization electrons
UV scintillation photons (~175 nm)

Figure 18: Diagram of the dual phase Xenon TPC in the LUX experiment which consists of two arrays of PMTs.

Events of interest consist of the following kinematical steps:

1. Incident particles interact with the liquid Xenon (LXe) causing either a **nuclear or electron recoil event**. (Results are insensitive to the underlying mediator particle).
2. The recoil event releases photons **(S1)** and ionization electrons.
3. An electric field causes the electrons to **drift towards the gas/liquid interface**.

The Liquid Underground Xenon (LUX) experiment

LUX is a dual phase (gas/liquid) Xenon TPC (time projection chamber) direct detection experiment searching for dark matter.

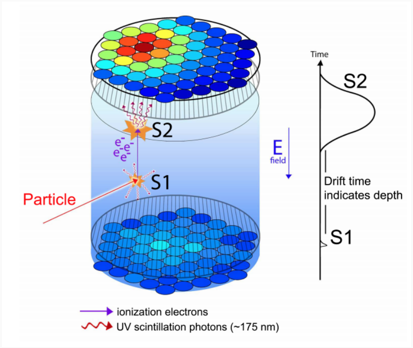

Time
S2
S2
E
field
Drift time indicates depth
S1
S1
Particle
ionization electrons
UV scintillation photons (~175 nm)

Figure 18: Diagram of the dual phase Xenon TPC in the LUX experiment which consists of two arrays of PMTs.

Events of interest consist of the following kinematical steps:

1. Incident particles interact with the liquid Xenon (LXe) causing either a **nuclear or electron recoil event**. (Results are insensitive to the underlying mediator particle).
2. The recoil event releases photons **(S1)** and ionization electrons.
3. An electric field causes the electrons to **drift towards the gas/liquid interface**.
4. The electrons cause additional scintillation events at the gas/liquid interface which releases photons **(S2)**.

The Liquid Underground Xenon (LUX) experiment

LUX is a dual phase (gas/liquid) Xenon TPC (time projection chamber) direct detection experiment searching for dark matter.

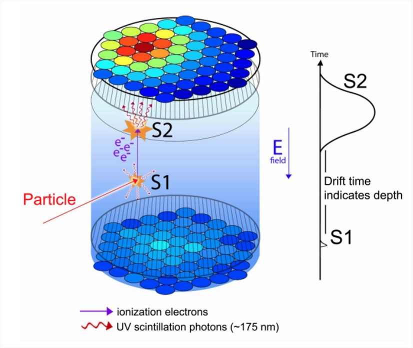


Figure 18: Diagram of the dual phase Xenon TPC in the LUX experiment which consists of two arrays of PMTs.

Events of interest consist of the following kinematical steps:

1. Incident particles interact with the liquid Xenon (LXe) causing either a **nuclear or electron recoil event**. (Results are insensitive to the underlying mediator particle).
2. The recoil event releases photons **(S1)** and ionization electrons.
3. An electric field causes the electrons to **drift towards the gas/liquid interface**.
4. The electrons cause additional scintillation events at the gas/liquid interface which releases photons **(S2)**.
5. Both the (S1) and (S2) photons are detected by the PMTs.

RQs (reduced quantities)

- Electron recoil (ER) - background events.
- Nuclear recoil (NR) - signal events.

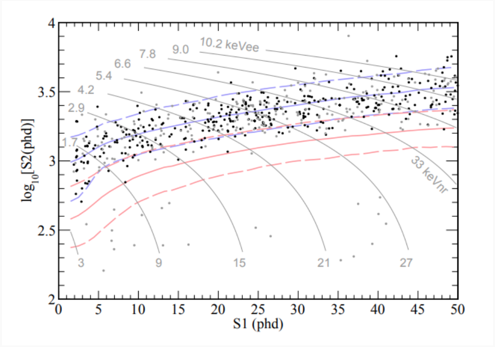


Figure 19: Observed events in the 2013 LUX exposure of 95 live days and 145 kg fiducial mass. Distributions of uniform in-energy electron recoils (blue) and an example 50 GeV c^2 WIMP signal (red)[25].

Classification sets limits on the WIMP-nucleon cross-section.

The low-level variables consist of 122 PMT signals. Typically one uses the following high-level (reduced quantities) statistics:

- **(S1)** - Sum of the number of photons detected (phd) under the initial scintillation pulse.
- **(S2)** - Sum of the number of phd under the secondary scintillation pulse.
- **(r,z)** - Reconstructed position of the event within the detector.

25 LUX Collaboration, Improved Limits on Scattering of Weakly Interacting Massive Particles from Reanalysis of 2013 LUX data, Phys. Rev. Lett. 116, 161301 (2016)

The Profile Likelihood Ratio Method

The profile-likelihood ratio (PLR) method is a maximum likelihood with additional steps.

The Profile Likelihood Ratio Method

The profile-likelihood ratio (PLR) method is a maximum likelihood with additional steps.

There are two sets of parameters:

The Profile Likelihood Ratio Method

The profile-likelihood ratio (PLR) method is a maximum likelihood with additional steps.

There are two sets of parameters:

- **Parameters of interest** (POI) - In our case the POI is the number of WIMP signal events n_{WS}.

The Profile Likelihood Ratio Method

The profile-likelihood ratio (PLR) method is a maximum likelihood with additional steps.

There are two sets of parameters:

- **Parameters of interest** (POI) - In our case the POI is the number of WIMP signal events n_{WS}.
- **Nuisance parameters** - Various detector dependent parameters and background rates λ.

Parameter	Type
n_{WS}	POI
g_1	nuisance
$g_{2,DD}$	nuisance
$k_{\mathrm{Lind.}}$	nuisance
n^{β}_{ER}	nuisance
$\vdots$	$\vdots$

Table 3: Various nuisance parameters for analysis on LUX.

The Profile Likelihood Ratio Method

The profile-likelihood ratio (PLR) method is a maximum likelihood with additional steps.

There are two sets of parameters:

- **Parameters of interest** (POI) - In our case the POI is the number of WIMP signal events n_{WS}.
- **Nuisance parameters** - Various detector dependent parameters and background rates λ.

Parameter	Type
n_{WS}	POI
g_1	nuisance
$g_{2,DD}$	nuisance
$k_{\mathrm{Lind.}}$	nuisance
n_{ER}^{β}	nuisance
$\vdots$	$\vdots$

Table 3: Various nuisance parameters for analysis on LUX.

We want to fit the distribution $p(x|n_{\mathrm{WS}}, \lambda)$ where $x = (S1, S2, r, z)$.

There are **several difficulties**:

The Profile Likelihood Ratio Method

The profile-likelihood ratio (PLR) method is a maximum likelihood with additional steps.

There are two sets of parameters:

- **Parameters of interest** (POI) - In our case the POI is the number of WIMP signal events n_{WS}.
- **Nuisance parameters** - Various detector dependent parameters and background rates λ.

Parameter	Type
n_{WS}	POI
g_1	nuisance
$g_{2,DD}$	nuisance
$k_{\mathrm{Lind.}}$	nuisance
n^{β}_{ER}	nuisance
$\vdots$	$\vdots$

Table 3: Various nuisance parameters for analysis on LUX.

We want to fit the distribution $p(x|n_{\mathrm{WS}}, \lambda)$ where $x = (S1, S2, r, z)$.

There are **several difficulties**:

1. Curse of dimensionality - For spaces X for which $\dim(X) > 3$ and the number of bins is large, the PLR method becomes computationally intensive. (Limited by $\#$ bins $\times$ $\#$ variables).

The Profile Likelihood Ratio Method

The profile-likelihood ratio (PLR) method is a maximum likelihood with additional steps.

There are two sets of parameters:

- **Parameters of interest** (POI) - In our case the POI is the number of WIMP signal events n_{WS}.
- **Nuisance parameters** - Various detector dependent parameters and background rates λ.

Parameter	Type
n_{WS}	POI
g_1	nuisance
$g_{2,DD}$	nuisance
$k_{\mathrm{Lind.}}$	nuisance
n^{β}_{ER}	nuisance
$\vdots$	$\vdots$

Table 3: Various nuisance parameters for analysis on LUX.

We want to fit the distribution $p(x|n_{\mathrm{WS}}, \lambda)$ where $x = (S1, S2, r, z)$.

There are **several difficulties**:

1. Curse of dimensionality - For spaces X for which $\dim(X) > 3$ and the number of bins is large, the PLR method becomes computationally intensive. (Limited by $\#$ bins $\times$ $\#$ variables).
2. Assumptions of independence - To deal with this, often we make assumptions about the independence between variable subsets, e.g.

$$(S1, S2, r, z) \rightarrow (S1, S2) \times (r, z) \quad (8)$$

N.B. This necessarily destroys information (correlations)!

The ML/PLR Method

To fix these issues, we first transform X using a neural network so that

$$\mathrm{X} \to f(\mathrm{X}) \quad \Rightarrow \quad p(n_{\mathrm{WS}}, \lambda | x) = p(n_{\mathrm{WS}}, \lambda | f(x)). \tag{9}$$

The ML/PLR Method

To fix these issues, we first transform X using a neural network so that

$$\mathrm{X} \rightarrow f(\mathsf{X}) \quad \Rightarrow \quad p(n_{\mathrm{WS}}, \lambda | x) = p(n_{\mathrm{WS}}, \lambda | f(x)). \tag{9}$$

The benefits of doing this include:

The ML/PLR Method

To fix these issues, we first transform X using a neural network so that

$$\mathrm{X} \to f(\mathrm{X}) \quad \Rightarrow \quad p(n_{\mathrm{WS}}, \lambda | x) = p(n_{\mathrm{WS}}, \lambda | f(x)). \tag{9}$$

The benefits of doing this include:

1. By checking that $f(\mathrm{X})$ satisfies $I[f(\mathrm{X}); \theta] = I[\mathrm{X}; \theta]$, all **correlations are preserved** within X.

The ML/PLR Method

To fix these issues, we first transform X using a neural network so that

$$\mathrm{X} \to f(\mathsf{X}) \quad \Rightarrow \quad p(n_{\mathrm{WS}}, \lambda | x) = p(n_{\mathrm{WS}}, \lambda | f(x)). \tag{9}$$

The benefits of doing this include:

1. By checking that $f(\mathsf{X})$ satisfies $I[f(\mathsf{X});\theta] = I[\mathsf{X};\theta]$, all **correlations are preserved** within X.
2. The input space to the PLR is simplified to a **single variable**.

The ML/PLR Method

To fix these issues, we first transform X using a neural network so that

$$\mathrm{X} \to f(\mathrm{X}) \quad \Rightarrow \quad p(n_{\mathrm{WS}}, \lambda | x) = p(n_{\mathrm{WS}}, \lambda | f(x)). \tag{9}$$

The benefits of doing this include:

1. By checking that $f(\mathrm{X})$ satisfies $I[f(\mathrm{X}); \theta] = I[\mathrm{X}; \theta]$, all **correlations are preserved** within X.
2. The input space to the PLR is simplified to a **single variable**.
3. Once a neural network model is generated, the **computational cost of running the PLR is drastically reduced** (by at least an order of magnitude!) – $\sim 600 \to 60$ minutes.

The ML/PLR Method

To fix these issues, we first transform X using a neural network so that

$$\mathrm{X} \rightarrow f(\mathrm{X}) \quad \Rightarrow \quad p(n_{\mathrm{WS}}, \lambda | x) = p(n_{\mathrm{WS}}, \lambda | f(x)). \tag{9}$$

The benefits of doing this include:

1. By checking that $f(\mathrm{X})$ satisfies $I[f(\mathrm{X}); \theta] = I[\mathrm{X}; \theta]$, all **correlations are preserved** within X.
2. The input space to the PLR is simplified to a **single variable**.
3. Once a neural network model is generated, the **computational cost of running the PLR is drastically reduced** (by at least an order of magnitude!) – $\sim 600 \rightarrow 60$ minutes.
4. One can **add additional variables** to X (e.g. pulse shape) **without increasing the computational complexity** of the PLR!

Preliminary Results for Run03

Using simulated events from the LUX Run03 re-analysis (2013)[25], we

1. Constructed signal data sets for a range of WIMP masses between 3.5 – 1000GeV.
2. Calculated the **upper-limit** on the set of input variables $(S1, S2, r, z)$ for each mass.
3. Trained an **ensemble** of neural networks on each mass.
4. Using the MI score of the neural network output, **chose the best model** out of the ensemble for each mass.

Below is an example of a neural network output for a 50GeV WIMP signal, together with the data from Run03.

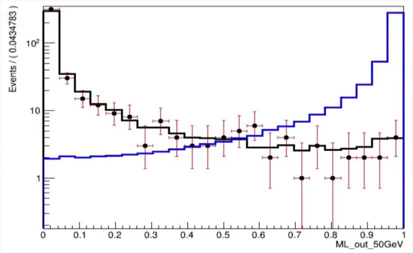


Figure 20: Histogram of the output of a neural network trained on simulation for Run03 where the signal corresponds to a 50 GeV WIMP. The black data points represent the live data from Run03.

25 LUX Collaboration, Improved Limits on Scattering of Weakly Interacting Massive Particles from Reanalysis of 2013 LUX data, Phys. Rev. Lett. 116, 161301 (2016)

Preliminary Results cont.

Using the best model output from each ensemble, we then run the PLR on the model output with the following parameters:

Parameter	Type
n_{WS}	POI
n_{ER}	nuisance

N.B. We get an identical result to the traditional approach with,

Preliminary Results cont.

Using the best model output from each ensemble, we then run the PLR on the model output with the following parameters:

Parameter	Type
n_{WS}	POI
n_{ER}	nuisance

N.B. We get an identical result to the traditional approach with,

1. an **order of magnitude lower** computational time – ($\sim 600 \rightarrow 60$ min.), and

Preliminary Results cont.

Using the best model output from each ensemble, we then run the PLR on the model output with the following parameters:

Parameter	Type
n_{WS}	POI
n_{ER}	nuisance

N.B. We get an identical result to the traditional approach with,

1. an **order of magnitude lower** computational time – ($\sim$ 600 $\rightarrow$ 60 min.), and
2. we can add additional variables without increasing computational complexity (e.g. pulse shape)[26,4].

Preliminary Results cont.

Using the best model output from each ensemble, we then run the PLR on the model output with the following parameters:

Parameter	Type
n_{WS}	POI
n_{ER}	nuisance

N.B. We get an identical result to the traditional approach with,

1. an **order of magnitude lower** computational time – ($\sim 600 \rightarrow 60$ min.), and
2. we can add additional variables without increasing computational complexity (e.g. pulse shape)[26,4].

With only the single nuisance parameter (number of background events), we find the following limit compared to the original Run03 re-analysis[4]:

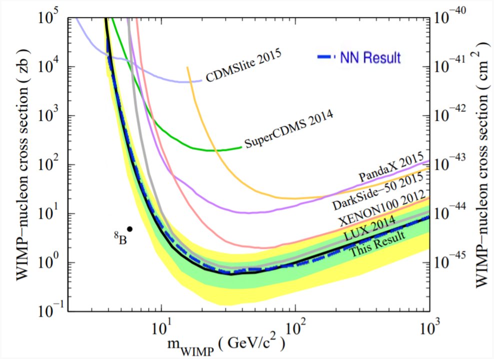


Figure 21: PLR limits on the neural network output for Run03[4].

26 The LUX Collaboration, “Liquid xenon scintillation measurements and pulse shape discrimination in the LUX dark matter detector”, https://arxiv.org/abs/1802.06162

4 The LUX Collaboration, “Combining Machine Learning and Profile Likelihood Methods on the LUX Experiment”, *In preparation*

Thanks

I would like to thank the physics department at the University at Albany as well as,

my advisor Jesse Ernst,

Ariel Caticha, Selman Ipek, Vivek Jain, Oleg Lunin and Kevin Vanslette for useful discussions regarding this work,

Scott Kravitz, Matthew Szydagis, Greg Rischbeiter and the LUX Collaboration.

Appendix

Estimating Mutual Information

Consider the problem of *estimating* (MI) from a sample distribution of N points,

$$\mathcal{X} \times \mathcal{Y} \subset \mathsf{X} \times \mathsf{Y}, \tag{10}$$

where,

$$\mathcal{X} \times \mathcal{Y} = \{x_i, y_i\}_{i=1}^{N}. \tag{11}$$

Given a sample which is i.i.d, we make the assumption that the local geometry of a point $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ approximately represents the probability density $p(x_i, y_i)$,

$$\begin{aligned} P(X,Y) &= \int_{X,Y} dxdy\, p(x,y) \\ &\approx c_d \varepsilon^d p(x_i, y_i), \end{aligned} \tag{12}$$

where $c_d \varepsilon^d$ represents a uniform "ball" of radius ε^d.

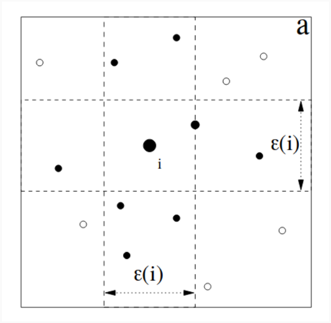
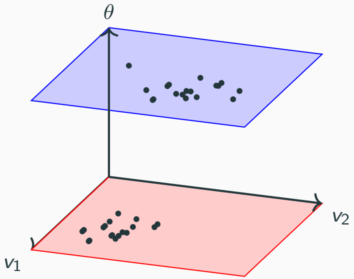


Figure 22: L^{∞} box around (x_i, y_i) to nearest neighbor.[6]

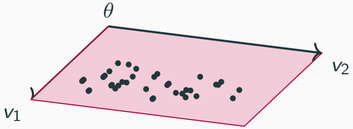
6 Kraskov, A., Stögbauer, H. and Grassberger, P., *Estimating Mutual Information*, https://arxiv.org/abs/cond-mat/0305641

Nearest Neighbor Mismatch

The KSG estimator essentially counts the number of nearest neighbor mismatches between the joint space and the marginal spaces



First, find the k-nearest neighbor distances $\Delta\bar{x}_i^k$ for all points $\bar{x}_i$ in the joint space (X, Θ).



Then, in each marginal space (X shown above) count the number of neighbors, n_{x_i}, within the distance $\Delta\bar{x}_i^k$ for each x_i.

Practical HEP example: SUSY dataset

We investigated the SUSY dataset[11] from the UCI MLR[12].

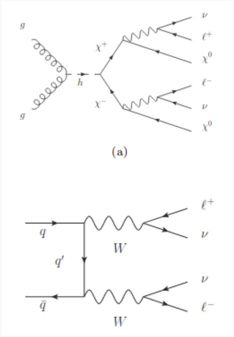


Figure 23: Signal $((\chi^+\chi^-) \to (\chi^0 W \chi^0 W) \to (\chi^0\chi^0)(\ell^+\ell^-\nu\nu))$ and background $((WW) \to (\ell^+\ell^-\nu\nu))$ processes for a SUSY search.

The dataset consists of **8** low-level kinematic variables and **10** high-level derived variables.

Low-level	High-level
p_T (lepton 1)	E_T^{rel} (missing energy)
η (lepton 1)	E_T^{axial} (missing energy)
ϕ (lepton 1)	M_R (razor)[13]
p_T (lepton 2)	$(M_R^T)^2$ (razor)
η (lepton 2)	R (razor)
ϕ (lepton 2)	$(M_T)^2$ (razor)
p_T (missing)	$\sqrt{\hat{s}_R}$ (razor)
ϕ (missing)	M_Δ^R (razor)
	$\Delta\phi_R^\beta$ (razor)
	$\cos(\theta_R + 1)$ (razor)

Masses for the supersymmetric particles are $m_{\chi^\pm} = 200\text{GeV}$ and $m_{\chi^0} = 100\text{GeV}$.

11 Baldi et. al., Searching for Exotic Particles in High-Energy Physics with Deep Learning, Nature Communications volume 5, 4308 (2014)

12 SUSY Data set, `https://archive.ics.uci.edu/ml/datasets/SUSY`

13 Buckley et. al., Super-Razor and Searches for Sleptons and Charginos at the LHC, Phys. Rev. D 89, 055020 (2014)