Physics and AI

Mike Douglas, Discussion at Strings 2021

[1]CMSA Harvard/Stony Brook

June 28, 2021

A bit of personal history

When I was deciding where to go to graduate school in 1983, I was torn between fundamental physics and artificial intelligence. In those days fundamental physics largely meant particle physics, and I had been advised by CN Yang that hard times were coming in that field.

So I went to Caltech and began my studies with John Hopfield, on his then-new neural network model, with the idea that I could go back to fundamental physics if developments warranted it. I also worked with Gerry Sussman (this was his first sabbatical year at Caltech).

In fall 1984 I switched to string theory and got my degree with John Schwarz in 1988. But I kept following AI and neural networks ever since. I also did ML for quantitative finance at Rentec from 2012-2020.

So, I will tell you about a few of the many ways in which physics influenced (and influences) machine learning, and then a bit about the other direction (much more about that from Fabian Ruehle).

A bit of personal history

When I was deciding where to go to graduate school in 1983, I was torn between fundamental physics and artificial intelligence. In those days fundamental physics largely meant particle physics, and I had been advised by CN Yang that hard times were coming in that field.

So I went to Caltech and began my studies with John Hopfield, on his then-new neural network model, with the idea that I could go back to fundamental physics if developments warranted it. I also worked with Gerry Sussman (this was his first sabbatical year at Caltech).

In fall 1984 I switched to string theory and got my degree with John Schwarz in 1988. But I kept following AI and neural networks ever since. I also did ML for quantitative finance at Rentec from 2012-2020.

So, I will tell you about a few of the many ways in which physics influenced (and influences) machine learning, and then a bit about the other direction (much more about that from Fabian Ruehle).

A bit of personal history

When I was deciding where to go to graduate school in 1983, I was torn between fundamental physics and artificial intelligence. In those days fundamental physics largely meant particle physics, and I had been advised by CN Yang that hard times were coming in that field.

So I went to Caltech and began my studies with John Hopfield, on his then-new neural network model, with the idea that I could go back to fundamental physics if developments warranted it. I also worked with Gerry Sussman (this was his first sabbatical year at Caltech).

In fall 1984 I switched to string theory and got my degree with John Schwarz in 1988. But I kept following AI and neural networks ever since. I also did ML for quantitative finance at Rentec from 2012-2020.

So, I will tell you about a few of the many ways in which physics influenced (and influences) machine learning, and then a bit about the other direction (much more about that from Fabian Ruehle).

A bit of personal history

When I was deciding where to go to graduate school in 1983, I was torn between fundamental physics and artificial intelligence. In those days fundamental physics largely meant particle physics, and I had been advised by CN Yang that hard times were coming in that field.

So I went to Caltech and began my studies with John Hopfield, on his then-new neural network model, with the idea that I could go back to fundamental physics if developments warranted it. I also worked with Gerry Sussman (this was his first sabbatical year at Caltech).

In fall 1984 I switched to string theory and got my degree with John Schwarz in 1988. But I kept following AI and neural networks ever since. I also did ML for quantitative finance at Rentec from 2012-2020.

So, I will tell you about a few of the many ways in which physics influenced (and influences) machine learning, and then a bit about the other direction (much more about that from Fabian Ruehle).

A bit of general history

Physicists were influential in the early development of many subfields of biology, and physicists' early involvement in AI was similar – think of the brain as a physical system; invent and analyze simplified models. As an example, the Hopfield model is essentially a spin glass,

$$H = -\sum_{i<j} J_{ij} S^i S^j$$

It can be used as a model of memory by postulating a dynamics (e.g. gradient descent) which takes an initial configuration $\vec{S}_{init}$ to a related $\vec{S}_{final}$ similar to one of a list of "memories" $\vec{S}_a$. This will work if we take

$$J_{ij} = \sum_{a=1}^{M} S_a^i S_a^j$$

and the number of memories $M \lesssim 0.1 N_{spins}$.

A bit of general history

During the 80's, the physics approach merged with other approaches to "natural" computing, most notably the perceptron (Rosenblatt 1957), the ancestor of the feed forward network (FFN).

- 1980's-90's – neural networks and physics based analyses, early applications of FFN's such as LeNet (Convnet for digit recognition). Symbolic AI – computer algebra, expert systems.
- 1990's – AI winter. Growing recognition that ML is a subfield of statistics – in physics, see Balasubramanian, cond-mat/9601030. Phase transitions in combinatorial optimization.
- 2000's – Systematic use of simple ML (SVM's, kernel methods).
- 2010's – Golden age of ML based on feed forward networks: AlexNet in 2012, AlphaGo in 2016 and Bert/GPT-2 in 2018.

Just as for string theory, many ideas in AI and ML went through an initial phase of great creativity and excitement, and then a phase of being deeply unfashionable, until convincing breakthroughs were made.

Standard ML tasks

- Supervised learning $\sim$ function fitting. Given data points (x_a, y_a) learn the "best" $y_a = F_W[x_a]$ from a parameterized space of functions F_W. (The parameters W are often called "weights".)
  For example, classifying images into human defined classes (say cat, dog, airplane, ...). Here $x \in \mathbb{R}^{M\times N}$ and y is a vector whose k'th component is the probability that the image is in class k.
- Probabilistic modeling or "self-supervised" learning. Given data points x_a learn the probability distribution $P_W[x]$ from a parameterized family which best fits the empirical distribution. For example, the distribution of events in a collider experiment (tracks, energy deposition, ...). Compare $P_W[x]$ from Standard Model plus detector simulation with observed $P_W[x]$.
- Reinforcement learning – choose actions guided by infrequent rewards. For example, learning to play a two-player game.

Standard ML tasks

- Supervised learning $\sim$ function fitting. Given data points (x_a, y_a) learn the "best" $y_a = F_W[x_a]$ from a parameterized space of functions F_W. (The parameters W are often called "weights".) For example, classifying images into human defined classes (say cat, dog, airplane, ...). Here $x \in \mathbb{R}^{M \times N}$ and y is a vector whose k'th component is the probability that the image is in class k.
- Probabilistic modeling or "self-supervised" learning. Given data points x_a learn the probability distribution $P_W[x]$ from a parameterized family which best fits the empirical distribution. For example, the distribution of events in a collider experiment (tracks, energy deposition, ...). Compare $P_W[x]$ from Standard Model plus detector simulation with observed $P_W[x]$.
- Reinforcement learning – choose actions guided by infrequent rewards. For example, learning to play a two-player game.

Standard ML tasks

- Supervised learning $\sim$ function fitting. Given data points (x_a, y_a) learn the "best" $y_a = F_W[x_a]$ from a parameterized space of functions F_W. (The parameters W are often called "weights".) For example, classifying images into human defined classes (say cat, dog, airplane, ...). Here $x \in \mathbb{R}^{M\times N}$ and y is a vector whose k'th component is the probability that the image is in class k.
- Probabilistic modeling or "self-supervised" learning. Given data points x_a learn the probability distribution $P_W[x]$ from a parameterized family which best fits the empirical distribution. For example, the distribution of events in a collider experiment (tracks, energy deposition, ...). Compare $P_W[x]$ from Standard Model plus detector simulation with observed $P_W[x]$.
- Reinforcement learning – choose actions guided by infrequent rewards. For example, learning to play a two-player game.

Standard ML tasks

- Supervised learning $\sim$ function fitting. Given data points (x_a, y_a) learn the "best" $y_a = F_W[x_a]$ from a parameterized space of functions F_W. (The parameters W are often called "weights".) For example, classifying images into human defined classes (say cat, dog, airplane, ...). Here $x \in \mathbb{R}^{M\times N}$ and y is a vector whose k'th component is the probability that the image is in class k.
- Probabilistic modeling or "self-supervised" learning. Given data points x_a learn the probability distribution $P_W[x]$ from a parameterized family which best fits the empirical distribution. For example, the distribution of events in a collider experiment (tracks, energy deposition, ...). Compare $P_W[x]$ from Standard Model plus detector simulation with observed $P_W[x]$.
- Reinforcement learning – choose actions guided by infrequent rewards. For example, learning to play a two-player game.

Standard ML tasks

- Supervised learning $\sim$ function fitting. Given data points (x_a, y_a) learn the "best" $y_a = F_W[x_a]$ from a parameterized space of functions F_W. (The parameters W are often called "weights".) For example, classifying images into human defined classes (say cat, dog, airplane, ...). Here $x \in \mathbb{R}^{M \times N}$ and y is a vector whose k'th component is the probability that the image is in class k.
- Probabilistic modeling or "self-supervised" learning. Given data points x_a learn the probability distribution $P_W[x]$ from a parameterized family which best fits the empirical distribution. For example, the distribution of events in a collider experiment (tracks, energy deposition, ...). Compare $P_W[x]$ from Standard Model plus detector simulation with observed $P_W[x]$.
- Reinforcement learning – choose actions guided by infrequent rewards. For example, learning to play a two-player game.

Standard ML models

The feed-forward network (FFN) or multilayer perceptron (MLP) maps an input vector space V_0 to an output vector space V_d. It is defined as an alternating composion of two simple functions, parameterized linear maps (matrices) $W^{(i)}$ from V_{i-1} to V_i, and a fixed nonlinear map θ,

$$F_W = W^{(d)} \circ \theta|_{V_{d-1}} \circ W^{(d-1)} \circ \ldots \circ \theta|_{V_2} \circ W^{(2)} \circ \theta|_{V_1} \circ W^{(1)}.$$

A standard choice for θ is the "ReLU" function applied componentwise,

$$\theta_{ReLU}(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases} \quad ; \qquad \theta_V \left(\sum_i c_i \, e_i \right) = \sum_i \theta(c_i) \, e_i .$$

One can show (Cybenko 1989) that even for two layers ($d = 2$), for sufficiently large dimension of the intermediate space V_1, this is a universal function approximator. As the depth d increases, the number of units needed to do this decreases.

To use the FFN for supervised learning, one defines an objective (or "loss") function which measures the error in the function approximation. For example, suppose data item x_a is in class k_a, then we could take

$$\mathcal{L} = -\sum_a P\left[F_W[x_a]\right]_{k_a}; \qquad P[f]_k \equiv \frac{\exp f_k}{\sum_{k'=1}^{M} \exp f_{k'}}.$$

The ratio of exponentials $P[\ldots]$ (often called softmax) maps $F_W \in \mathbb{R}^M$ to a vector of probabilities.

We then find good weights W by numerically minimizing $\mathcal{L}$ on a training dataset (with known k_a). Usually one does this by (stochastic) gradient descent starting from a random initial value W_0.

There are many variations ("architectures") used to adapt to known properties ("priors") of the data set and the task. For example, the convolutional neural network is good for a translationally invariant input distribution. Many generalizations to other symmetry structures have been proposed by physicists, see Cheng *et al* arXiv:1906.02481.

To use the FFN for supervised learning, one defines an objective (or "loss") function which measures the error in the function approximation. For example, suppose data item x_a is in class k_a, then we could take

$$\mathcal{L} = -\sum_a P\left[F_W[x_a]\right]_{k_a} ; \qquad P[f]_k \equiv \frac{\exp f_k}{\sum_{k'=1}^M \exp f_{k'}} .$$

The ratio of exponentials $P[\ldots]$ (often called softmax) maps $F_W \in \mathbb{R}^M$ to a vector of probabilities.

We then find good weights W by numerically minimizing $\mathcal{L}$ on a training dataset (with known k_a). Usually one does this by (stochastic) gradient descent starting from a random initial value W_0.

There are many variations ("architectures") used to adapt to known properties ("priors") of the data set and the task. For example, the convolutional neural network is good for a translationally invariant input distribution. Many generalizations to other symmetry structures have been proposed by physicists, see Cheng *et al* arXiv:1906.02481.

Important ML research questions

The vast bulk of ML research is practical – code efficient ML software, develop new datasets and benchmark tasks, try out architectures and optimization schemes, tune parameters to improve performance.

Some theoretical questions of widely accepted importance have emerged. The most important is to explain how FNN's generalize, *i.e.* deal correctly with inputs similar to but not literally in the training set. In statistics one explains this in terms of models which encode priors. Are there more general mechanisms?

There is a more specific paradox which arises from the fact that successful FFN's usually have large numbers of weights (millions, billions), so many that they can completely fit the training set. The traditional dogma of statistics says that such a model will "overfit" and generalize poorly. This was a major reason for the belief (widely held before 2012) that deep networks would not work.

Important ML research questions

The vast bulk of ML research is practical – code efficient ML software, develop new datasets and benchmark tasks, try out architectures and optimization schemes, tune parameters to improve performance.

Some theoretical questions of widely accepted importance have emerged. The most important is to explain how FNN's generalize, *i.e.* deal correctly with inputs similar to but not literally in the training set. In statistics one explains this in terms of models which encode priors. Are there more general mechanisms?

There is a more specific paradox which arises from the fact that successful FFN's usually have large numbers of weights (millions, billions), so many that they can completely fit the training set. The traditional dogma of statistics says that such a model will "overfit" and generalize poorly. This was a major reason for the belief (widely held before 2012) that deep networks would not work.

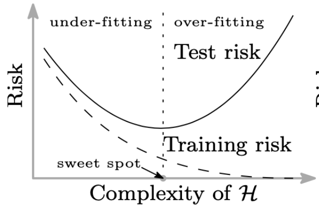
Important ML research questions

The vast bulk of ML research is practical – code efficient ML software, develop new datasets and benchmark tasks, try out architectures and optimization schemes, tune parameters to improve performance.

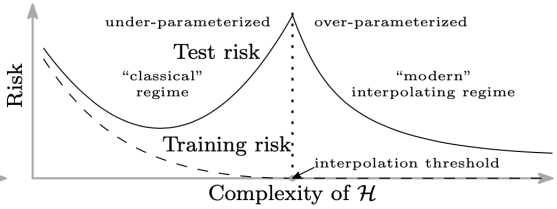
Some theoretical questions of widely accepted importance have emerged. The most important is to explain how FNN's generalize, *i.e.* deal correctly with inputs similar to but not literally in the training set. In statistics one explains this in terms of models which encode priors. Are there more general mechanisms?

There is a more specific paradox which arises from the fact that successful FFN's usually have large numbers of weights (millions, billions), so many that they can completely fit the training set. The traditional dogma of statistics says that such a model will "overfit" and generalize poorly. This was a major reason for the belief (widely held before 2012) that deep networks would not work.

The most popular resolution of the paradox is that the initial conditions and optimization methods used in practice lead to "implicit regularization" which decreases the effective number of weights. However this is not universally accepted and the details are not clear.



(a) U-shaped "bias-variance" risk curve



(b) "double descent" risk curve

Figure 1: Curves for training risk (dashed line) and test risk (solid line). (a) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (b) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high complexity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

The double descent curve, from Belkin *et al*, arXiv:1812.11118.

What can physics do for ML and AI ?

How can one do theory in such a data-driven field? In statistics, one focuses on simple and universal distributions. This was the standard approach in ML during the 1990's – people developed generative models which approximated real world distributions of images and other inputs. However such models are very complicated and do not perform as well as more general FFN's trained on large datasets.

Still there can be universal properties of models. A major source for them uses the analogy to disordered systems in physics. For example, the Hopfield model with independently drawn random couplings J_{ij} shares properties of the models with "real" memories, such as the threshold for the number of memories it can learn, $M \lesssim 0.1 N_{spins}$.

Physics intuition suggests that this threshold will become sharp in the limit of a large model $N_{spins} \to \infty$ and many memories $M \to \infty$. Doing this with the ratio $\alpha = M/N_{spins}$ fixed, there is a phase transition: for $\alpha < \alpha_c$ the model can learn all the memories, for $\alpha > \alpha_c$ none of them.

What can physics do for ML and AI ?

How can one do theory in such a data-driven field? In statistics, one focuses on simple and universal distributions. This was the standard approach in ML during the 1990's – people developed generative models which approximated real world distributions of images and other inputs. However such models are very complicated and do not perform as well as more general FFN's trained on large datasets.

Still there can be universal properties of models. A major source for them uses the analogy to disordered systems in physics. For example, the Hopfield model with independently drawn random couplings J_{ij} shares properties of the models with "real" memories, such as the threshold for the number of memories it can learn, $M \lesssim 0.1 N_{spins}$.

Physics intuition suggests that this threshold will become sharp in the limit of a large model $N_{spins} \to \infty$ and many memories $M \to \infty$. Doing this with the ratio $\alpha = M/N_{spins}$ fixed, there is a phase transition: for $\alpha < \alpha_c$ the model can learn all the memories, for $\alpha > \alpha_c$ none of them.

What can physics do for ML and AI ?

How can one do theory in such a data-driven field? In statistics, one focuses on simple and universal distributions. This was the standard approach in ML during the 1990's – people developed generative models which approximated real world distributions of images and other inputs. However such models are very complicated and do not perform as well as more general FFN's trained on large datasets.

Still there can be universal properties of models. A major source for them uses the analogy to disordered systems in physics. For example, the Hopfield model with independently drawn random couplings J_{ij} shares properties of the models with "real" memories, such as the threshold for the number of memories it can learn, $M \lesssim 0.1 N_{spins}$.

Physics intuition suggests that this threshold will become sharp in the limit of a large model $N_{spins} \to \infty$ and many memories $M \to \infty$. Doing this with the ratio $\alpha = M/N_{spins}$ fixed, there is a phase transition: for $\alpha < \alpha_c$ the model can learn all the memories, for $\alpha > \alpha_c$ none of them.

Phase transitions appear in many problems in machine learning, and theoretical results were first obtained by physicists using the replica method from spin glass theory. See the textbook by Mézard and Montanari (2009), *Information, Physics, and Computation*.

Another general method to produce synthetic data is the "teacher-student" setup. Here one could take two FFN's, one a teacher F_{W_T} with random weights W_T which generates the input-output relation. We then train a student FFN F_{W_S} on data pairs x_a random and $y_a = F_{W_T}(x_a)$. See Zdeborová and Krzakala arXiv:1511.02476 for many applications of statistical physics methods in this context.

Random matrix theory is also widely applied in ML and statistics. One source of its relevance is that the initial conditions for FFN weights are often taken to be random normal variables, leading to a standard matrix ensemble. A recent tour de force work which derives the double descent curve using free probability methods is Adlam and Pennington arXiv:2008.06786.

Phase transitions appear in many problems in machine learning, and theoretical results were first obtained by physicists using the replica method from spin glass theory. See the textbook by Mézard and Montanari (2009), *Information, Physics, and Computation*.

Another general method to produce synthetic data is the "teacher-student" setup. Here one could take two FFN's, one a teacher F_{W_T} with random weights W_T which generates the input-output relation. We then train a student FFN F_{W_S} on data pairs x_a random and $y_a = F_{W_T}(x_a)$. See Zdeborová and Krzakala arXiv:1511.02476 for many applications of statistical physics methods in this context.

Random matrix theory is also widely applied in ML and statistics. One source of its relevance is that the initial conditions for FFN weights are often taken to be random normal variables, leading to a standard matrix ensemble. A recent tour de force work which derives the double descent curve using free probability methods is Adlam and Pennington arXiv:2008.06786.

Phase transitions appear in many problems in machine learning, and theoretical results were first obtained by physicists using the replica method from spin glass theory. See the textbook by Mézard and Montanari (2009), *Information, Physics, and Computation*.

Another general method to produce synthetic data is the "teacher-student" setup. Here one could take two FFN's, one a teacher F_{W_T} with random weights W_T which generates the input-output relation. We then train a student FFN F_{W_S} on data pairs x_a random and $y_a = F_{W_T}(x_a)$. See Zdeborová and Krzakala arXiv:1511.02476 for many applications of statistical physics methods in this context.

Random matrix theory is also widely applied in ML and statistics. One source of its relevance is that the initial conditions for FFN weights are often taken to be random normal variables, leading to a standard matrix ensemble. A recent tour de force work which derives the double descent curve using free probability methods is Adlam and Pennington arXiv:2008.06786.

What can ML and AI do for physics?

Many experts predict that artificial general intelligence will be created within 30–40 years. That equals the span of my own career, from the Green-Schwarz anomaly cancellation up to now.

What about the coming decade? ML for science is very active, mostly data driven but some *ab initio*. Fabian will speak more on this.

What about conceptual influences?

- My own work on statistics of vacua was partly inspired by my study of statistics for ML. This evolved into the String Vacuum Project, a forerunner of the current ML for string theory work.
- Use statistical concepts (KL divergence, information geometry) for spaces of QFT, *e.g.* Balasubramanian *et al*, arXiv:1410.6809
- Maybe architecture of FFN's or other models has physical analogs. See for example Akutagawa *et al* arXiv:2005.02636 which develops the analogy: layer propagation $\sim$ RG $\sim$ AdS.

What can ML and AI do for physics?

Many experts predict that artificial general intelligence will be created within 30–40 years. That equals the span of my own career, from the Green-Schwarz anomaly cancellation up to now.

What about the coming decade? ML for science is very active, mostly data driven but some *ab initio*. Fabian will speak more on this.

What about conceptual influences?

- My own work on statistics of vacua was partly inspired by my study of statistics for ML. This evolved into the String Vacuum Project, a forerunner of the current ML for string theory work.
- Use statistical concepts (KL divergence, information geometry) for spaces of QFT, *e.g.* Balasubramanian *et al*, arXiv:1410.6809
- Maybe architecture of FFN's or other models has physical analogs. See for example Akutagawa *et al* arXiv:2005.02636 which develops the analogy: layer propagation $\sim$ RG $\sim$ AdS.

What can ML and AI do for physics?

Many experts predict that artificial general intelligence will be created within 30–40 years. That equals the span of my own career, from the Green-Schwarz anomaly cancellation up to now.

What about the coming decade? ML for science is very active, mostly data driven but some *ab initio*. Fabian will speak more on this.

What about conceptual influences?

- My own work on statistics of vacua was partly inspired by my study of statistics for ML. This evolved into the String Vacuum Project, a forerunner of the current ML for string theory work.
- Use statistical concepts (KL divergence, information geometry) for spaces of QFT, *e.g.* Balasubramanian *et al*, arXiv:1410.6809
- Maybe architecture of FFN's or other models has physical analogs. See for example Akutagawa *et al* arXiv:2005.02636 which develops the analogy: layer propagation $\sim$ RG $\sim$ AdS.

AI for theorem proving

In the last few years I have been looking into the use of AI to do mathematical reasoning, both as a test of AI and potentially to support research in the mathematical sciences.

Computers can verify mathematical theorems expressed in formal terms, *i.e.* such that every logical step can be made explicitly. But despite great progress in developing interactive theorem proving systems such as Lean and Coq, they are hard to learn and use.

Since about 2017 several AI groups (at Google, OpenAI, ...) have been trying to use ML to solve this problem. There is an analogy between steps in a proof and moves in a game of solitaire, a problem which can be attacked using reinforcement learning. Another approach uses language models (Bert, GPT, ...) to predict the next steps of a proof.

Could we develop a search engine which, given a precise description of a mathematical concept, finds related documents?

AI for theorem proving

In the last few years I have been looking into the use of AI to do mathematical reasoning, both as a test of AI and potentially to support research in the mathematical sciences.

Computers can verify mathematical theorems expressed in formal terms, *i.e.* such that every logical step can be made explicitly. But despite great progress in developing interactive theorem proving systems such as Lean and Coq, they are hard to learn and use.

Since about 2017 several AI groups (at Google, OpenAI, ...) have been trying to use ML to solve this problem. There is an analogy between steps in a proof and moves in a game of solitaire, a problem which can be attacked using reinforcement learning. Another approach uses language models (Bert, GPT, ...) to predict the next steps of a proof.

Could we develop a search engine which, given a precise description of a mathematical concept, finds related documents?

AI for theorem proving

In the last few years I have been looking into the use of AI to do mathematical reasoning, both as a test of AI and potentially to support research in the mathematical sciences.

Computers can verify mathematical theorems expressed in formal terms, *i.e.* such that every logical step can be made explicitly. But despite great progress in developing interactive theorem proving systems such as Lean and Coq, they are hard to learn and use.

Since about 2017 several AI groups (at Google, OpenAI, ...) have been trying to use ML to solve this problem. There is an analogy between steps in a proof and moves in a game of solitaire, a problem which can be attacked using reinforcement learning. Another approach uses language models (Bert, GPT, ...) to predict the next steps of a proof.

Could we develop a search engine which, given a precise description of a mathematical concept, finds related documents?

AI for theorem proving

In the last few years I have been looking into the use of AI to do mathematical reasoning, both as a test of AI and potentially to support research in the mathematical sciences.

Computers can verify mathematical theorems expressed in formal terms, *i.e.* such that every logical step can be made explicitly. But despite great progress in developing interactive theorem proving systems such as Lean and Coq, they are hard to learn and use.

Since about 2017 several AI groups (at Google, OpenAI, ...) have been trying to use ML to solve this problem. There is an analogy between steps in a proof and moves in a game of solitaire, a problem which can be attacked using reinforcement learning. Another approach uses language models (Bert, GPT, ...) to predict the next steps of a proof.

Could we develop a search engine which, given a precise description of a mathematical concept, finds related documents?

Fundamental roles of complexity and intelligence?

Physics puts limits on computation, for example erasing a bit costs $kT/2$ free energy. What does the theory of computation have to say about fundamental physics? Aaronson quant-ph/0502072: physical systems cannot do NP hard computations efficiently. Many recent ideas about quantum informational underpinnings of space-time.

What does the theory of computation say about cosmology and vacuum selection? Work with Denef, Greene and Zukowski:

- CCL I (hep-th/0602072) – Finding a vacuum which tunes away the cosmological constant is NP hard in string theory.
- CCL II (1706.06430) – Reformulates vacuum selection as a search problem. Even the hard tuning discussed in CCL I can be done far more efficiently by a computer than by eternal inflation with its doubly exponentially small tunneling rates. Almost any possible scenario could be more likely than eternal inflation.

Related ideas in Khoury *et al* 1907.07693, Alexander *et al* 2104.03902.

Fundamental roles of complexity and intelligence?

Physics puts limits on computation, for example erasing a bit costs $kT/2$ free energy. What does the theory of computation have to say about fundamental physics? Aaronson quant-ph/0502072: physical systems cannot do NP hard computations efficiently. Many recent ideas about quantum informational underpinnings of space-time.

What does the theory of computation say about cosmology and vacuum selection? Work with Denef, Greene and Zukowski:

- CCL I (hep-th/0602072) – Finding a vacuum which tunes away the cosmological constant is NP hard in string theory.
- CCL II (1706.06430) – Reformulates vacuum selection as a search problem. Even the hard tuning discussed in CCL I can be done far more efficiently by a computer than by eternal inflation with its doubly exponentially small tunneling rates. Almost any possible scenario could be more likely than eternal inflation.

Related ideas in Khoury *et al* 1907.07693, Alexander *et al* 2104.03902.