Modern Computational Economics and Policy Applications

A Workshop at the IMF

Chase Coleman and John Stachurski

2nd December 2025

Introduction

Introductory slides cover

- Background: advanced in AI
- Coding with AI
- AI mania — impact on hardware and software
- Consequences for economists

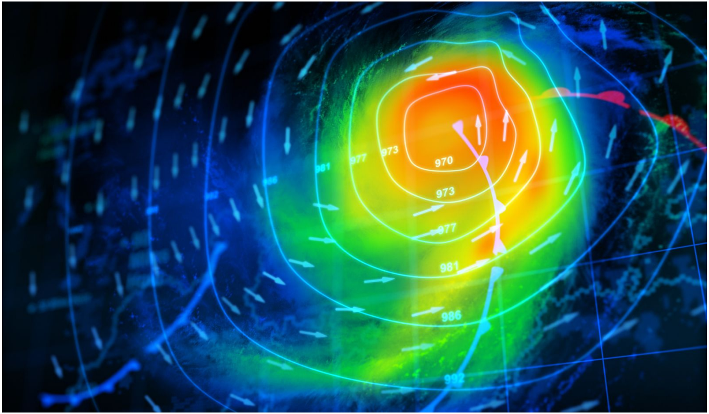
Background: Progress in AI

- image processing / computer vision
- translation
- forecasting and prediction
- generative AI (LLMs, image / music / video)
- etc.

Image Generators

Video Generators

Forecasting



"ECMWF's model is considered the gold standard for medium-term weather forecasting…Google DeepMind claims to beat it 90% of the time…"

— MIT Technology Review 2024

"Google's GenCast model outdid the ECMWF forecasts 97.2 percent of the time when predicting 1,320 global atmospheric features"

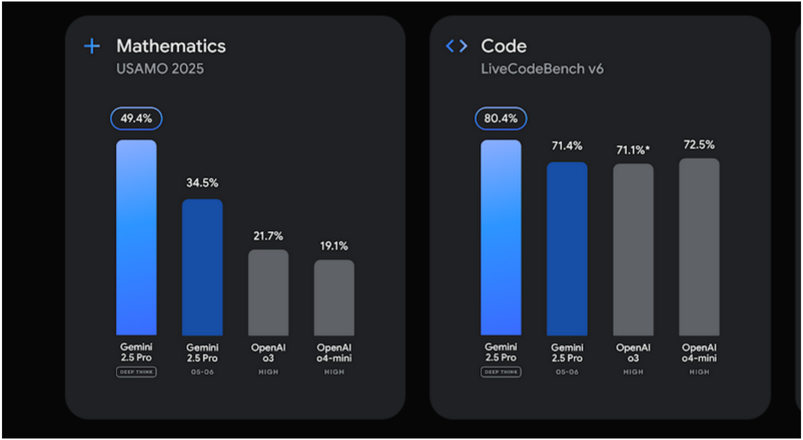
— Weatherstats 2025

"Traditional forecasting models are big, complex computer algorithms [that] take hours to run. AI models can create forecasts in just seconds."

Also successful in predicting

- electricity prices
- renewable energy supply
- fraudulent transations
- patient admission rates in hospitals
- sales and demand (e-commerce)
- traffic flow
- delivery times
- etc.

LLMs


Mathematics
USAMO 2025
49.4%
34.5%
21.7%
19.1%
Gemini 2.5 Pro
DEEP THINK
Gemini 2.5 Pro
05-06
OpenAI o3
HIGH
OpenAI o4-mini
HIGH
Code
LiveCodeBench v6
80.4%
71.4%
71.1%*
72.5%
Gemini 2.5 Pro
DEEP THINK
Gemini 2.5 Pro
05-06
OpenAI o3
HIGH
OpenAI o4-mini
HIGH

Claude: "Post-training RL is absolutely crucial for my reasoning abilities."

- Pre-trained model (trained only on next-token prediction) can do some reasoning, but it's inconsistent and often produces meandering or incorrect logical chains.

Process:

- Human raters evaluate my outputs for logical coherence
- A reward model is trained to predict these human preferences
- Training process uses RL via proximal policy optimization

Learns to recognize and reject logical errors

Example: Coding with AI

```
ɸ john on xps …/papers/opt_at_point/presentations/grips on β main [✗→?]
❯❯ claude

        Claude Code v2.0.21
        Sonnet 4.5 · Claude Pro
        /home/john/gh_synced/papers/opt_at_point/presentations/grips

> write code for a dynamic inventory problem that will be solved with value function
  iteration and (for testing purposes) Howard policy iteration.  a firm holds
  inventory for and sells a single good.  consumer demand is stochastic and iid,
  following a geometric distribution.  the price of the good is also iid and follows
  a lognormal distribution.  firms cannot sell more than the inventory they have on
  hand.

                                                     ctrl-g to edit prompt in nvim

[13] [0:grips]                                           Oct 17 13:25 | john@xps
```

AI coding affects optimal language choice

Claude Sonnet 4.5:

“I’m definitely stronger with Python than MATLAB.”

“My capabilities with Python are more comprehensive. I have deeper familiarity with Python’s extensive ecosystem of libraries, frameworks, and modern development practices.”

“I can more confidently help with advanced Python topics, debugging complex Python code, and implementing Python best practices.”

"I'm definitely stronger with Python than Julia."

"Python is one of my most proficient languages - I have deep familiarity with its syntax, libraries, frameworks, and best practices across many domains including data science, web development, machine learning, and general-purpose programming."

"While I understand Julia's syntax and core concepts, my expertise with it isn't as comprehensive as with Python."

Claude:

Thank you! It was a great exercise working through this model. We accomplished quite a lot:

- Fixed critical bugs
- Improved architecture
- Enhanced code quality

Your suggestions throughout - especially making K global for JAX compatibility and using the builder pattern for the Model - really improved the overall design!

Pros and cons

- Doesn't see the big picture
- Can ace small tasks but struggles to connect them
- You still need to be the architect
- Sometimes AI gets weird

Example: AlphaEvolve

Google Deepmind May 2025

A coding agent for scientific and algorithmic discovery

- Employs an evolutionary algorithm
- Asks an ensemble of LLMs and then iterates, tests, refines

Process

1. Proposed solutions evaluated
2. Promising solutions are selected and mutated by LLMs
3. "Survival of the fittest" progressively improves performance

Outcomes at Google:

- Enhanced efficiency in chip design (TPUs)
- Improved data center scheduling
- Discovered new matrix multiplication algorithms (surpassing Strassen's algorithm for 4x4 complex matrices)

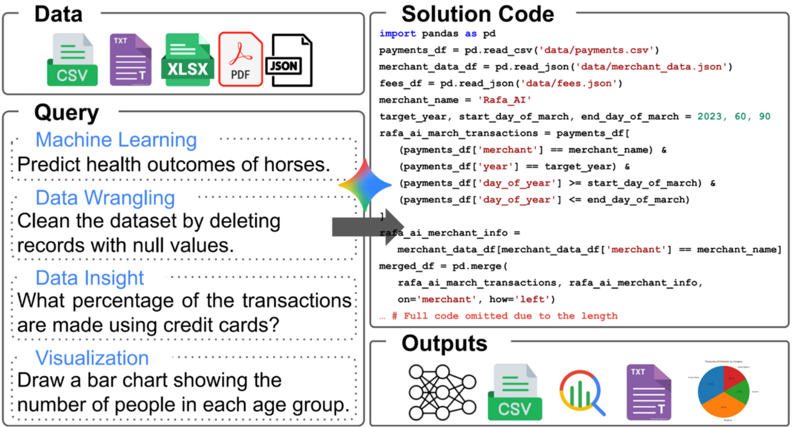
Example: DS-STAR

A coding agent for automated data science (Google Research)

A data science agent that automates tasks from statistical analysis to visualization across various data types

"Transforms raw data into actionable insights" by automating document interpretation and statistical analysis.

- Built on LLMs
- "engages in a loop of planning, implementing, and verifying."
- Achieves top performance on the DABStep benchmark.


Data
CSV
TXT
XLSX
PDF
JSON
Query
Machine Learning
Predict health outcomes of horses.
Data Wrangling
Clean the dataset by deleting records with null values.
Data Insight
What percentage of the transactions are made using credit cards?
Visualization
Draw a bar chart showing the number of people in each age group.
Solution Code
import pandas as pd
payments_df = pd.read_csv('data/payments.csv')
merchant_data_df = pd.read_json('data/merchant_data.json')
fees_df = pd.read_json('data/fees.json')
merchant_name = 'Rafa_AI'
target_year, start_day_of_march, end_day_of_march = 2023, 60, 90
rafa_ai_march_transactions = payments_df[
(payments_df['merchant'] == merchant_name) &
(payments_df['year'] == target_year) &
(payments_df['day_of_year'] >= start_day_of_march) &
(payments_df['day_of_year'] <= end_day_of_march)
]
rafa_ai_merchant_info =
merchant_data_df[merchant_data_df['merchant'] == merchant_name]
merged_df = pd.merge(
rafa_ai_march_transactions, rafa_ai_merchant_info,
on='merchant', how='left')
… # Full code omitted due to the length
Outputs
CSV
TXT

Investment

Private AI investment by US firms in 2024 = \$109 billion USD

Estimate for 2025 = \$615 billion

Massive investments in

- data centers
- server / GPU / TPU design and production
- software development

Deep Learning

All of the AI projects listed above use **deep learning**

- Representation of relationships through “artificial neural networks”
- The networks are “trained” through gradient descent

As a result, understanding the foundations of DL can help us

- build and work with AI-adjacent models
- understand why hardware and software have evolved to the present state
- predict where these environments are heading

Whirlwind introduction to deep learning

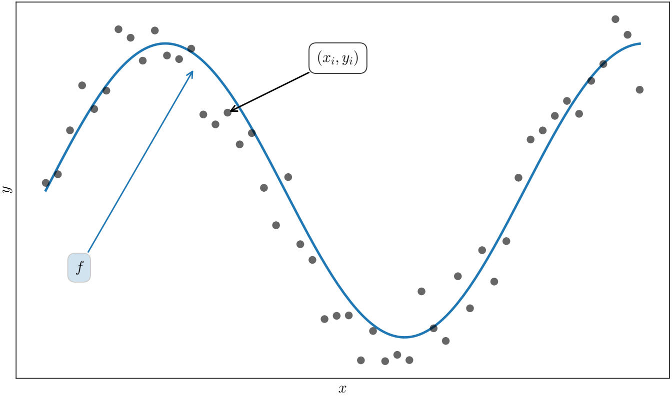
Let's start with the learning problem

We observe input-output pairs (x, y), where

- $x \in \mathbb{R}^k$
- $y \in \mathbb{R}$ (for example)

Egs.

- x = market indicators at t; y = prob of financial crisis at $t+1$
- x = electricity consumption at $t-s, \ldots, t$; y = demand at t



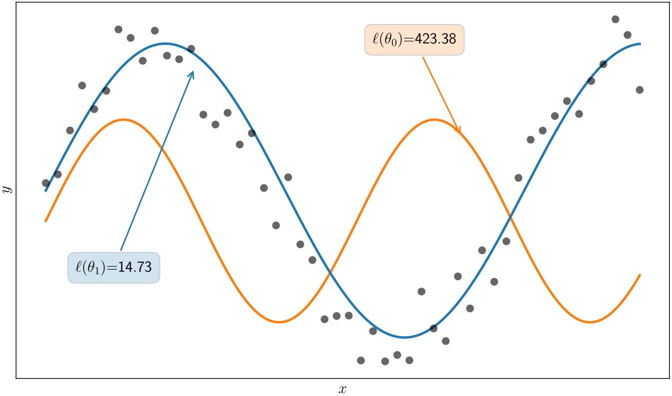
Problem: observe $(x_i, y_i)_{i=1}^n$ and seek f such that $y_{n+1} \approx f(x_{n+1})$

Nonlinear Regression

Training:

1. Choose function class $\{f_\theta\}_{\theta\in\Theta}$
2. Minimize loss

$$\ell(\theta) := \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2 \quad \text{s.t.} \quad \theta \in \Theta$$


$\ell(\theta_0)$=423.38
$\ell(\theta_1)$=14.73
y
x

Deep Learning (DL)

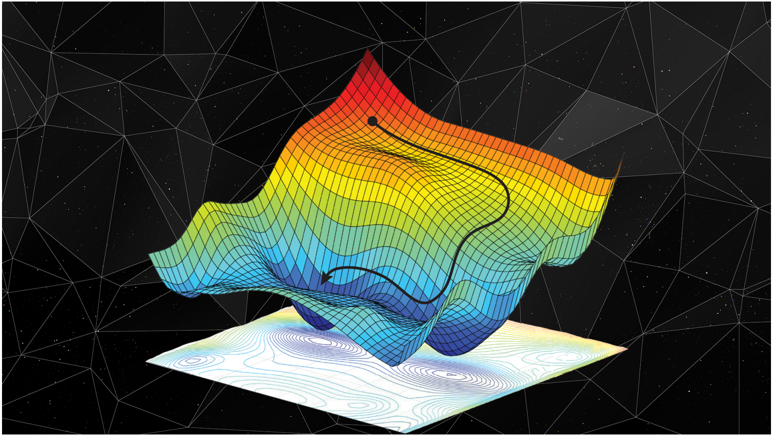
In the case of DL, elements of $\{f_\theta\}_{\theta\in\Theta}$ have a particular structure:

$$f_\theta = A_\theta^m \circ \boldsymbol{\sigma} \circ \cdots \circ A_\theta^2 \circ \boldsymbol{\sigma} \circ A_\theta^1$$

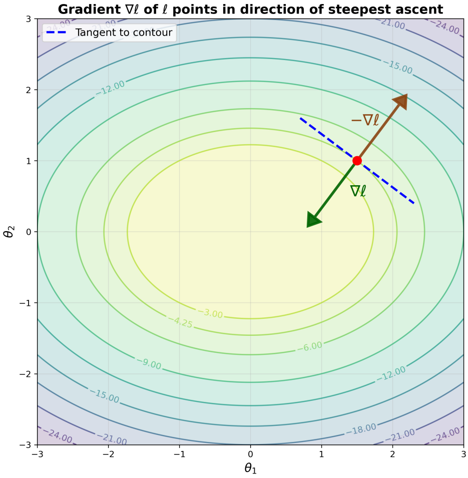
- each A_θ^i is an "affine" function
- σ is a "activation" function

Choosing θ means choosing the parameters in the neural net

Minimizing $\ell(\theta)$ – what algorithm?



Source: https://danielkhv.com/

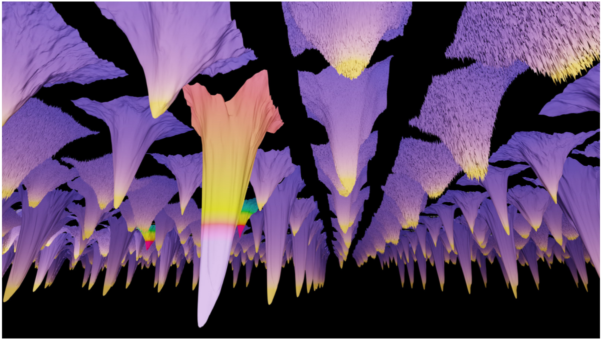

Gradient ∇ℓ of ℓ points in direction of steepest ascent
Tangent to contour
−∇ℓ
∇ℓ
θ1
θ2
−3.00
−4.25
−6.00
−9.00
−12.00
−15.00
−18.00
−21.00
−24.00

Gradient descent

$$\theta_{n+1} = \theta_n - \lambda \nabla \ell(\theta_n)$$

- θ_n = current guess
- λ = learning rate
- $\nabla \ell$ = gradient of loss function

Deep learning: $\theta \in \mathbb{R}^d$ where $d =?$



Source: https://losslandscape.com/gallery/

How does it work?

How is it possible to minimize loss over such high dimensions??

Core elements

1. parallelization over powerful hardware (GPUs or TPUs)
2. automatic differentiation (for gradient descent)
3. Compilers / JIT-compilers for fast parallelized machine code

How does it work?

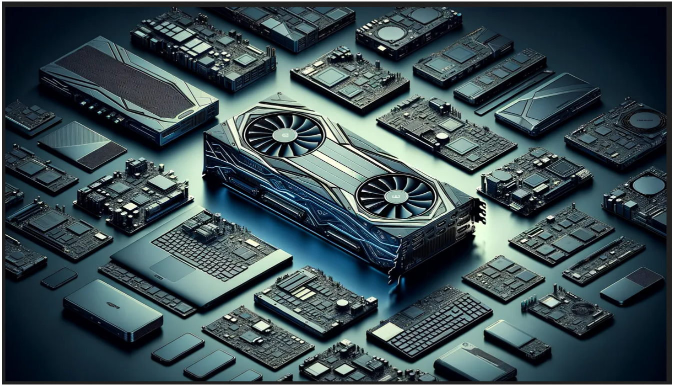
How is it possible to minimize loss over such high dimensions??

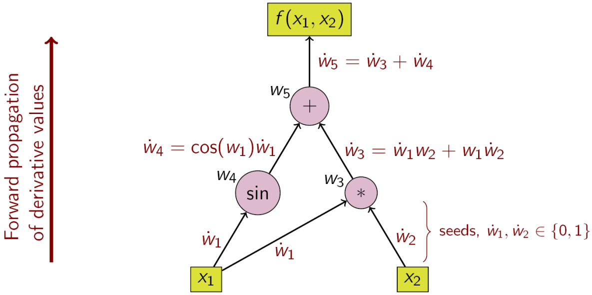
Core elements

1. parallelization over powerful hardware (GPUs or TPUs)
2. automatic differentiation (for gradient descent)
3. Compilers / JIT-compilers for fast parallelized machine code

Parallelization

Parallelization




Forward propagation
of derivative values
$f(x_1, x_2)$
$\dot{w}_5 = \dot{w}_3 + \dot{w}_4$
w_5
+
$\dot{w}_4 = \cos(w_1)\dot{w}_1$
$\dot{w}_3 = \dot{w}_1 w_2 + w_1 \dot{w}_2$
w_4
sin
w_3
*
$\dot{w}_1$
$\dot{w}_1$
$\dot{w}_2$
seeds, $\dot{w}_1, \dot{w}_2 \in \{0, 1\}$
x_1
x_2

Just-in-time compilers

```python
@jax.jit
def f(x):
    return jnp.sin(x) - jnp.cos(x**2)
```

- detects and adapts to problem dimensions
- detects and adapts to existing hardware
- automatic parallelization

Automatic differentiation

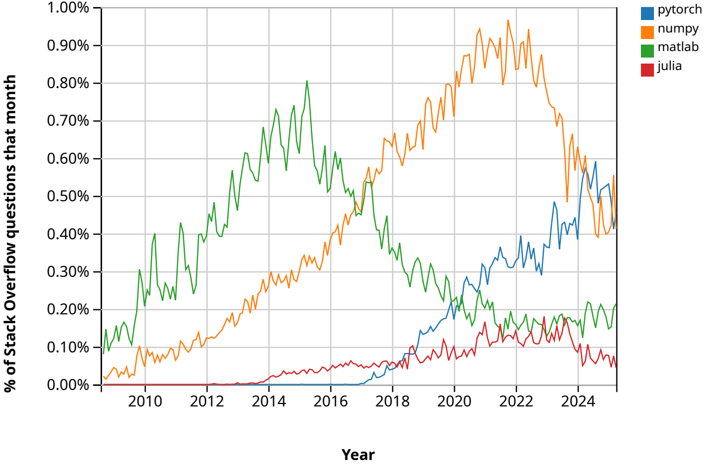
```
from jax import grad

def f(θ, x):
    # add details here
    return prediction

def loss(θ, x, y):
  return jnp.sum((y - f(θ, x))**2)

loss_gradient = grad(loss)   # exact automatic differentiation
θ = θ - λ * loss_gradient(θ, x_data, y_data)
```

Platforms

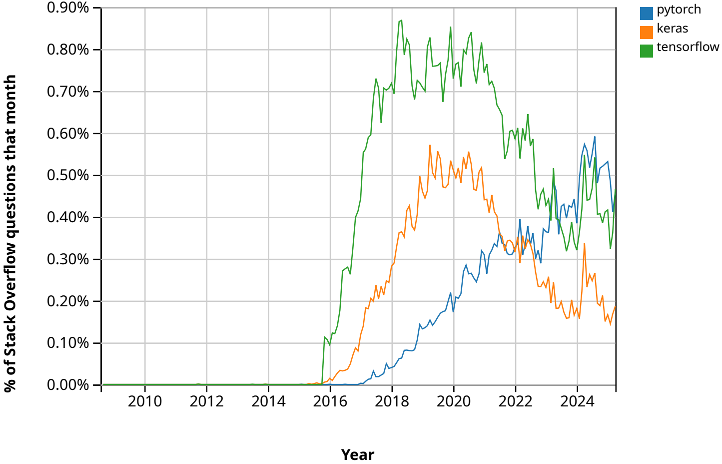
Platforms that support AI / deep learning:

- Tensorflow
- PyTorch (Llama, ChatGPT)
- Google JAX (Gemini, DeepMind)
- Keras (backends = JAX, PyTorch)
- Mojo (Modular (Python))
- MATLAB???

Popularity – languages and libraries



Popularity – DL / ML frameworks



AI tools for economic modeling

Let's say that you want to do computational economics without deep learning

Can these new AI tools be applied?

Yes! Yes! Yes!

- fast matrix algebra
- fast solutions to linear systems
- fast nonlinear system solvers
- fast optimization, etc.

AI tools for economic modeling

Let's say that you want to do computational economics without deep learning

Can these new AI tools be applied?

Yes! Yes! Yes!

- fast matrix algebra
- fast solutions to linear systems
- fast nonlinear system solvers
- fast optimization, etc.

Case Study

The CBC uses the "overborrowing" model of Bianchi (2011)

- credit constraint loosens during booms
- bad shocks $\longrightarrow$ sudden stops

CBC implementation in MATLAB

- runs on $10,000 mainframe with 356 CPUs and 1TB RAM
- runtime = 12 hours

Rewrite in Python + Google JAX

- runs on $400 gaming GPU with 10GB RAM
- runtime = 7 seconds

Case Study

The CBC uses the "overborrowing" model of Bianchi (2011)

- credit constraint loosens during booms
- bad shocks $\longrightarrow$ sudden stops

CBC implementation in MATLAB

- runs on $10,000 mainframe with 356 CPUs and 1TB RAM
- runtime = 12 hours

Rewrite in Python + Google JAX

- runs on $400 gaming GPU with 10GB RAM
- runtime = 7 seconds

Summary

- We are at the start of a massive AI revolution
- This revolution will have a huge impact on science
- What impact on economics?

Aims

- Better understanding of core AI methods
- Better understanding of core tools (hardware / software)
- Apply knowledge to current economic modeling

Summary

- We are at the start of a massive AI revolution
- This revolution will have a huge impact on science
- What impact on economics?

Aims

- Better understanding of core AI methods
- Better understanding of core tools (hardware / software)
- Apply knowledge to current economic modeling