Computational Economics and the AI Revolution

John Stachurski

2025

Topics

- Introduction
- ANNs and deep learning
- Stochastic approximation
- Dynamic programming and reinforcement learning

AI-driven scientific computing

AI is changing the world

- generative AI (LLMs, image / music / video generators)
- image processing / computer vision
- speech recognition, translation
- scientific knowledge discovery
- forecasting and prediction
- etc

Plus killer drones, Skynet, etc....

AI-driven scientific computing

AI is changing the world

- generative AI (LLMs, image / music / video generators)
- image processing / computer vision
- speech recognition, translation
- scientific knowledge discovery
- forecasting and prediction
- etc

Plus killer drones, Skynet, etc....

Private AI investment in 2024:

- U.S. = \$109 billion
- China \$9.3 billion
- UK \$4.5 billion

Massive investments in

- data centers
- GPUs
- software

Deep learning in two slides

Aim: approximate an unknown functional relationship

$$y = f(x) \qquad (x \in \mathbb{R}^k, \ y \in \mathbb{R})$$

Examples.

- $x =$ cross section of returns, $y =$ return on oil futures tomorrow
- $x =$ weather sensor data, $y =$ max temp tomorrow

Problem:

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

Nonlinear regression:

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

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

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

- We discuss this structure soon
- Typically, $\theta \mapsto f_\theta(x)$ is smooth for all x
- MSE is a popular loss function but others are also used

Nonlinear regression:

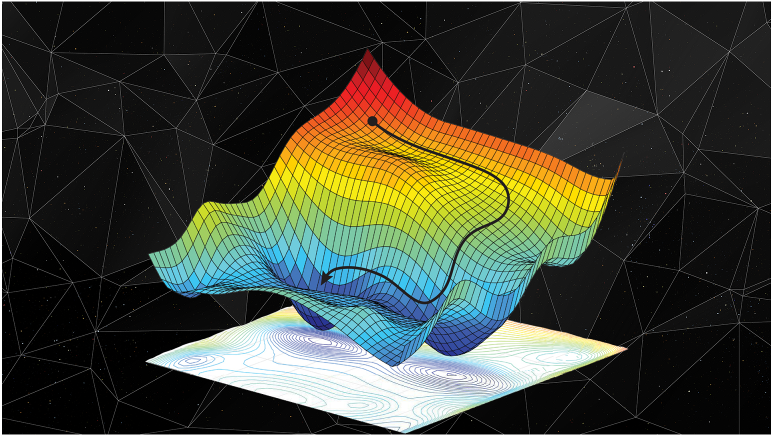
1. Choose function class $\{f_\theta\}_{\theta \in \Theta}$
2. Minimize the MSE

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

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

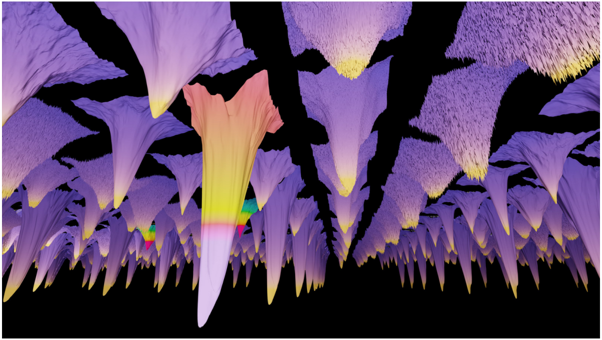
- We discuss this structure soon
- Typically, $\theta \mapsto f_\theta(x)$ is smooth for all x
- MSE is a popular loss function but others are also used

Minimizing a smooth loss functions – what algorithm?



Source: https://danielkhv.com/

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



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

How does it work?

Why is it possible to minimize over $\theta \in \mathbb{R}^d$ when $d = 10^{12}$?!?

Core elements

- automatic differentiation (for gradient descent)
- parallelization (GPUs or TPUs)
- Compilers / JIT-compilers

How does it work?

Why is it possible to minimize over $\theta \in \mathbb{R}^d$ when $d = 10^{12}$?!?

Core elements

- automatic differentiation (for <u>gradient</u> descent)
- parallelization (GPUs or TPUs)
- Compilers / JIT-compilers

Automatic differentiation

"Exact numerical" differentiation

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

loss_gradient = grad(loss)
```

Now use gradient descent...

Parallelization

```
outputs = pmap(f, data)
```

- multithreading over GPU cores (how many?)
- multiprocessing over accelerators in a GPU farm / supercomputing cluster (how many?)

Just-in-time compilers

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

Advantages over AOT compilers:

- cleaner code
- more portable
- automatic parallelization (same code for CPUs / GPUs)

Advantages over NumPy / MATLAB

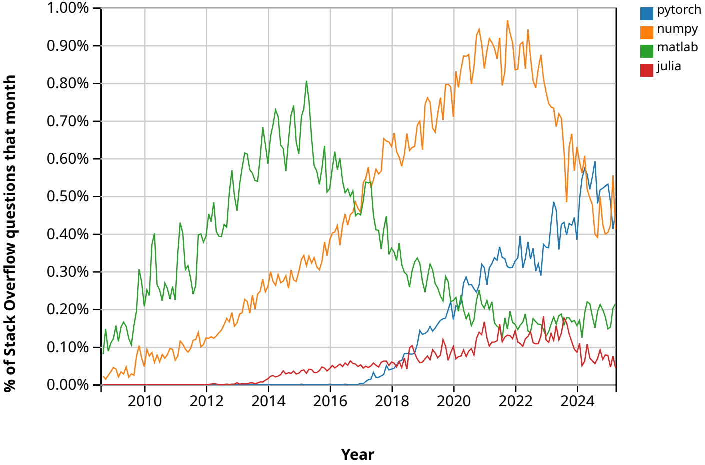
- can specialize machine code based on parameter types / shapes
- automatically matches tasks with accelerators (GPU / TPU)
- fuses array operations for speed and memory efficiency

Platforms

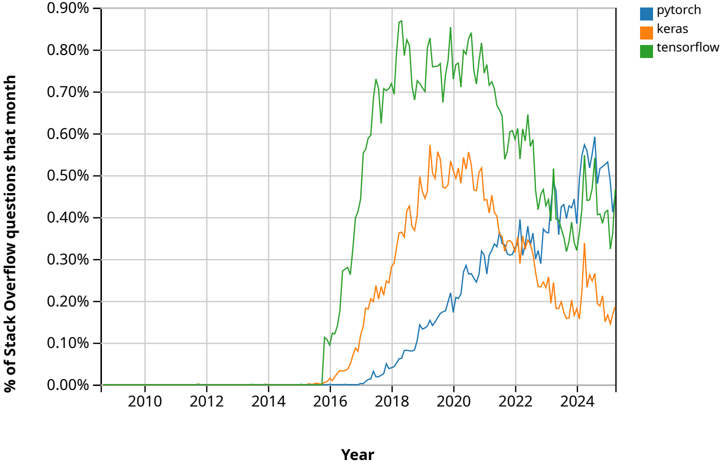
Platforms that support AI / deep learning:

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

Popularity – languages



Popularity – deep learning 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.

Advantages of JAX (vs PyTorch / Numba / etc.) for economists:

- exposes low level functions
- elegant functional programming style – close to maths
- elegant autodiff tools
- array operations follow standard NumPy API
- automatic parallelization
- same code, multiple backends (CPUs, GPUs, TPUs)

Case Study

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

- credit constraint loosens during booms
- bad shocks → 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 = 4.17 seconds

Case Study

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

- credit constraint loosens during booms
- bad shocks $\rightarrow$ 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 $=$ 4.17 seconds