Computational Economics and the AI Revolution

John Stachurski

2025

Topics

1. The AI revolution
2. ANNs and deep learning
3. AI software and hardware
4. Repurposing AI software and hardware for econ
5. Applications

The aim of this lecture is limited

- Better understanding of core AI methods
- Better understanding of core tools (hardware / software)
- How can this knowledge be applied to economic modeling today?

Focus is technical

- Build understanding of maths / stats / code

The AI revolution

- generative AI
- image processing / computer vision
- speech recognition
- translation
- scientific knowledge discovery
- forecasting and prediction
- etc.

Example: AlphaEvolve

A coding agent for scientific and algorithmic knowledge discovery

Google Deepmind (May 2025)

Develops algorithms and codebases using LLMs

Asks an ensemble of LLMs and then tests, iterates, refines

Employs an evolutionary algorithm

- Promising solutions are selected and mutated by LLMs
- "Survival of the fittest" progressively improves performance

Discovered a new matrix multiplication algorithm

- Surpasses Strassen's algorithm for 4x4 matrices
- Breaks a 56 year old record

Example: AlphaFold

- Developed by Google DeepMind
- Predicts 3D protein structures from string of amino acids

Outcomes

- Accelerating drug discovery and design
- Accelerating research on cancer / Alzheimer's / etc.
- Supporting enzyme engineering for sustainability

Authors awarded 2024 Nobel Prize in Chemistry

LLMs

J is john stachurski smart?

Based on what I know, John Stachurski is considered quite accomplished in his field. He's a professor of economics known for his work in economic theory, computational economics, and machine learning applications in economics.

He's particularly recognized for developing QuantEcon, an open-source project that provides resources for quantitative economics, which demonstrates both technical expertise and a commitment to education and open access to knowledge.

His contributions to economic modeling, particularly dynamic economic models, suggest strong technical and mathematical abilities. His published research and textbooks are well-regarded in the academic community.

Would you like me to search for more recent information about John Stachurski's work and contributions?

Retry

Claude can make mistakes. Please double-check responses.

J wouldn't you say he is very accomplished?

Yes, I would absolutely say John Stachurski is very accomplished. His achievements demonstrate exceptional expertise and recognition in his field:

- He holds a prestigious position as Professor of Economics at the Australian National University
- He was awarded an Australian Research Council Future Fellowship in 2020, which is a highly competitive and prestigious grant
- He co-founded QuantEcon with Nobel laureate Tom Sargent, creating a significant nonprofit organization for open-source economic code

Gemini

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

Google Veo 3

Weather forecasts

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

"Traditional forecasting models are big, complex computer algorithms based on atmospheric physics and take hours to run. AI models can create forecasts in just seconds."

Source: MIT Technology Review July 2024

AI for coding

J please write a function for gradient descent in 6 lines or less

Here's a gradient descent function in 6 lines:

```python
def gradient_descent(grad_func, init_params, lr=0.01, iters=100):
    params = init_params.copy()
    for _ in range(iters):
        gradient = grad_func(params)
        params = params - lr * gradient
        if np.linalg.norm(lr * gradient) < 1e-6: break
    return params
```

AI code generation is great…but not perfect

- AI doesn't see the big picture
- Aces small tasks but struggles to connect them
- You still need to be the architect

— `Lonely-Public2655`

AI coding affects optimal language choice

"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'm definitely stronger with Python than Julia."

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

Killer drones, Skynet, etc.

Investment

US private AI investment

- 2024: $109 billion
- 2025 estimate: $350 billion

Massive investments in

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

What kinds of problems are they trying to solve?

ANNs: A model of the human brain

– source: Dartmouth undergraduate journal of science

ANN representation: directed acyclic graph

w_{ij} is called a "weight"

$$y_j = \sum_i x_i w_{ij}$$

$$y_1 = \sum_i x_i w_{i1}$$

$$y_2 = \sum_i x_i w_{i2}$$

$$\implies y = xW$$

Next steps

Add bias:

$$y_j = \sum_i x_i w_{ij} \qquad \rightarrow \qquad y_j = \sum_i x_i w_{ij} + b_j$$

Apply activation:

$$y_j = \sum_i x_i w_{ij} + b_j \qquad \rightarrow \qquad y_j = \sigma\left(\sum_i x_i w_{ij} + b_j\right)$$

Applying σ pointwise, we can write this in vector form as

$$y = \sigma(xW + b)$$

Training

Training = adjusting the weights and biases to fit data

Aim: Learn to predict output y from input x

- $x \in \mathbb{R}^k$
- $y \in \mathbb{R}$ (regression problem)

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: choose model $\{f_\theta\}_{\theta\in\Theta}$ and minimize the empirical loss

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

In the case of ANNs, we consider all f_θ having the form

$$f_\theta = G_m \circ G_{m-1} \circ \cdots \circ G_2 \circ G_1$$

where

- $G_\ell x = \sigma_\ell(xW_\ell + b_\ell)$
- σ_ℓ is an activation function

Nonlinear regression: choose model $\{f_\theta\}_{\theta\in\Theta}$ and minimize the empirical loss

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

In the case of ANNs, we consider all f_θ having the form

$$f_\theta = G_m \circ G_{m-1} \circ \cdots \circ G_2 \circ G_1$$

where

- $G_\ell x = \sigma_\ell(xW_\ell + b_\ell)$
- σ_ℓ is an activation function

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/

Why does it work?

Good question

Why does it work?

Good question

How does it work?

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

1. Parallelization over powerful hardware
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??

1. Parallelization over powerful hardware
2. Automatic differentiation (for gradient descent)
3. Compilers / JIT-compilers for fast parallelized machine code

Parallelization

- Single-device parallelism

```
def function(data):
    # perform a calculation based on input data
    return output
vectorized_function = vmap(function)
# Perform the same action on a collection of data sets
outputs = vectorized_function(data_sets)
```

- Multi-device parallelism

```
parallel_function = pmap(function)
outputs = parallel_function(list_of_tasks)
```

Automatic differentiation

A systematic approach for computing derivatives of functions expressed as computer programs

Core idea:

1. Apply the chain rule of calculus to break down complex function evaluations into elementary operations
2. Compute derivatives by combining the derivatives of these elementary pieces

Outcome: Exact, efficient, automated

```python
# Set up loss function
def loss(θ, x, y):
  return jnp.sum((y - f(θ, x))**2)

# Set up gradient vector of loss function
loss_gradient = grad(loss)

# Evaluate gradient at θ
d = loss_gradient(θ, x_data, y_data)

# Update current guess
θ = θ - λ * d
```

Just-in-time compilers

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

Advantages over traditional (AOT) compilers:

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

Platforms

Platforms that support AI / deep learning:

- Tensorflow
- PyTorch (Llama, ChatGPT)
- Google JAX (Gemini, DeepMind)
- Keras (backends = JAX, PyTorch, Tensorflow)

Popularity – DL frameworks

Popularity – languages and libraries

I mainly use

- Developed by Google Research
- Rising popularity
- Elegant design that exposes low level routines

"The acronym JAX stands for Just After eXecution"

- monitor function execution once and then compile

Another acronym:

- Just-in-time compilation
- Automatic differentiation
- XLA (accelerated linear algebra)

Some typical JAX code

```
def f(θ, x):
  for W, b in θ:
    w = W @ x + b
    x = jnp.tanh(w)
  return x

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

grad_loss = jit(grad(loss))
```

AI tools for economic modeling

Is DL well-suited to economic modeling?

Example. Is macro-forecasting similar to weather forecasting?

Why or why not?

Repurposing AI tools

Let's say that you want to do standard macro modeling (not DL)

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.

Repurposing AI tools

Let's say that you want to do standard macro modeling (not DL)

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 $\rightarrow$ 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 = 7 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 $=$ 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?

At minimum, we can immediately use

- AI for code authoring
- AI hardware/software for standard computational econ models