Computational Economics Workshop

The University of Melbourne

John Stachurski

August 2024

Topics

Part 1: Workshop

- Current and future trends in scientific computing
- AI-driven scientific computing
- Applications (hands on, using Python)

Part 2: Computational Economics in Action

- Joint with James Hansen and Yong Song

Key questions:

- What computational skills should economists learn in 2024?
- What are some interesting applications?

Please feel free to question / debate / share your experiences

Flow

- 1:00 – 2:30 Lecture 1
- 2:30 – 3:00 afternoon tea (staff lounge)
- 3:00 – 4:00 Lecture 2
- 4:00 – 4:15 break
- 4:15 – 5:00 Computational Economics in Action

Slides, code:

https://github.com/QuantEcon/melbourne_2024

Quick poll:

- Python programmers?
  - NumPy? Numba? PyTorch? JAX?
- Julia?
- R?
- MATLAB?
- C?
- Fortran?

Regular GPU users?

Prelude: AI-driven scientific computing

AI is changing the world

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

Plus killer drones, skynet, etc....

Prelude: AI-driven scientific computing

AI is changing the world

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

Plus killer drones, skynet, etc....

Projected spending on AI in 2024:

- Google: $48 billion
- Microsoft: $60 billion
- Meta: $40 billion
- etc.

What kinds of problems are they solving?

Deep learning in two slides

Aim: find an approximation to an unknown functional relationship

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

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 $y_{n+1} \approx f(x_{n+1})$

Training: minimize the empirical loss

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

But what is $\{f_\theta\}_{\theta \in \Theta}$?

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

$$f_\theta = \sigma \circ A_1 \circ \cdots \circ \sigma \circ A_{k-1} \circ \sigma \circ A_k$$

where

- $A_i x = W_i x + b_i$ is an affine map
- σ is a nonlinear "activation" function

Training: minimize the empirical loss

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

But what is $\{f_\theta\}_{\theta\in\Theta}$?

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

$$f_\theta = \sigma \circ A_1 \circ \cdots \circ \sigma \circ A_{k-1} \circ \sigma \circ A_k$$

where

- $A_i x = W_i x + b_i$ is an affine map
- σ is a nonlinear "activation" function

Training: minimize the empirical loss

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

But what is $\{f_\theta\}_{\theta \in \Theta}$?

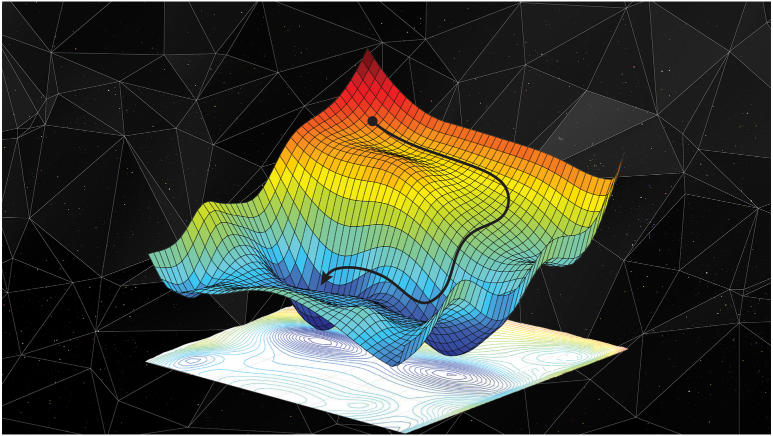
In the case of ANNs, we consider all f_θ having the form

$$f_\theta = \sigma \circ A_1 \circ \cdots \circ \sigma \circ A_{k-1} \circ \sigma \circ A_k$$

where

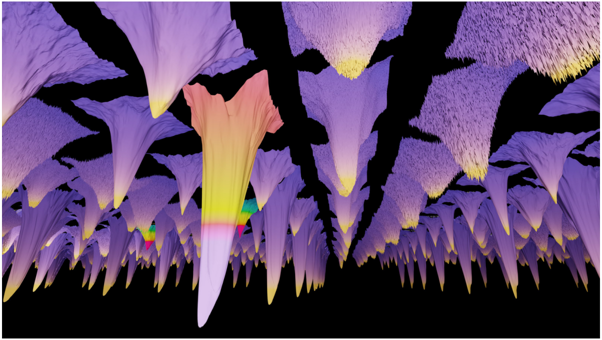
- $A_i x = W_i x + b_i$ is an affine map
- σ is a nonlinear "activation" function

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/

Hardware

"NVIDIA supercomputers are the factories of the AI industrial revolution." – Jensen Huang

Software

Core elements

- automatic differentiation (for gradient descent)
- parallelization (GPUs! — how many?)
- Compilers / JIT-compilers

Crucially, these components must be well integrated

Popular platforms with these features

- Pytorch
- Google JAX (with Python)

```python
import jax.numpy as jnp
from jax import grad, jit

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))  # Now use gradient descent
```

Source: JAX readthedocs

Summary

We can't afford the same hardware as OpenAI but we do have access to cheaper versions

And most of the relevent code is open source

Thus, AI-driven scientific computing is giving us many exciting and powerful new tools

- for deep learning <u>and</u> other kinds of mathematical modeling

Let's use them!