Python and the AI Revolution

Chase Coleman and John Stachurski

March 2024

Topics

We will discuss

- AI-driven scientific computing
- Where are we heading?
- Economic applications?

Sides

https://github.com/QuantEcon/imf_2024

AI-driven scientific computing

AI is changing the world

- deep learning / other machine learning
- large language models
- computer vision
- speech recognition
- scientific knowledge discovery
- forecasting and prediction, etc., etc.

The huge amount of resources being poured into AI is changing the choice set for **all** scientific coders

Key players

- OpenAI / Microsoft
- Google (Google Research, Google DeepMind)
- Meta
- Anthropic, etc.

Platforms / libraries

- PyTorch (ChatGPT, Meta's LLaMA 2, Stable Diffusion)
- Google JAX (Google's Gemini)
- Tensorflow, Keras, Mojo?

Lightening introduction to deep learning

Supervised deep learning: find a good approximation to an unknown functional relationship

$$y = f(x)$$

- x is the input and y is the output

Examples.

- $x =$ unfinished sentence, $y =$ next word
- $x =$ weather sensor data, $y =$ max temp tomorrow

Training

Nonlinear regression: Take data set $(x_i, y_i)_{i=1}^n$ and solve

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

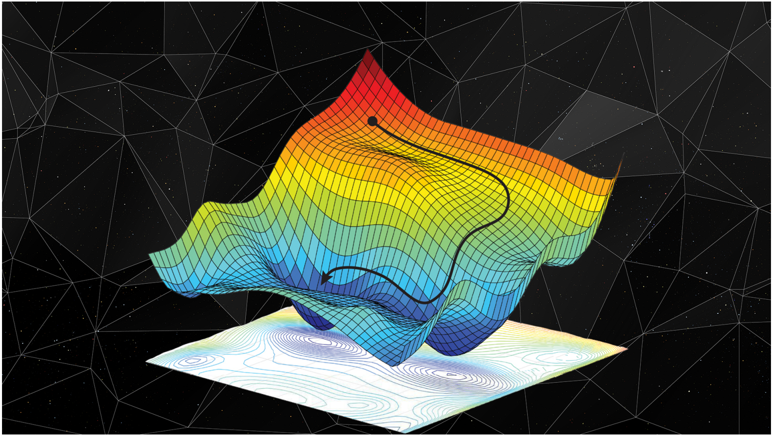
In the case of ANNs, we consider all f_θ having the form

$$f_\theta = \sigma \circ A_1 \circ \cdots \circ \sigma \circ A_k$$

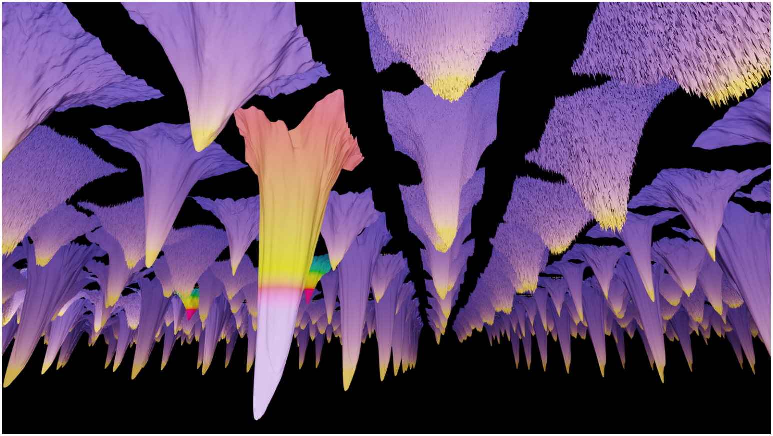
where

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

Minimizing a smooth loss functions



Source: https://danielkhv.com/



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

Core elements

- automatic differentiation!
- parallelization (CPUs / GPUs / TPUs)!
- Compilers / JIT-compilers!

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

def predict(params, x):
  for W, b in params:
    y = jnp.dot(W, x) + b
    x = jnp.tanh(y)
  return y

def loss(params, x, targets):
  preds = predict(params, x)
  return jnp.sum((preds - targets)**2)

grad_loss = jit(grad(loss))
# Now use gradient descent on the loss function
```

"ECMWF's weather forecasting model is considered the gold standard for medium-term weather forecasting...Google DeepMind claims in an non-peer-reviewed paper to have 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

Relevant to economics?

Deep learning provides massively powerful pattern recognition

But macroeconomic data is

- far more limited than weather observation sensor data
- generally nonstationary
- laws of motion change with policies (Lucas critique)

Possible applications:

- Finding stylized facts? Testing causal relationships?
- Numerical methods – approximating high-dimensional functions

Relevant to economics?

Deep learning provides massively powerful pattern recognition

But macroeconomic data is

- far more limited than weather observation sensor data
- generally nonstationary
- laws of motion change with policies (Lucas critique)

Possible applications:

- Finding stylized facts? Testing causal relationships?
- Numerical methods – approximating high-dimensional functions

One point of view

- Deep learning is not very relevant for policy-centric macroeconomic modeling
- Deep learning is yet to prove itself as a "better" approach to numerical methods
- And yet, at the same time, **the AI computing revolution is generating tools that are enormously beneficial for macroeconomic modeling**
    - autodiff, JIT compilers, parallelization, GPUs, etc.
- We can take full advantage of them right now...