Designing collective investment strategies with machine learning

John Armstrong Cristin Buescu James Dalby Rohan Hobbs

Department of Mathematics, King's College London

ICCF, 2026

Investment consumption problem

Economic inputs Black–Scholes market, $\mathrm{d}S_t = \mu S_t\,\mathrm{d}t + \sigma S_t\,\mathrm{d}W_t$. Deterministic income η_t before retirement at t_{RA}.
Controls: annually choose to invest fixed proportion π_t of wealth in risky asset, consume a proportion c_t.
Dynamics: Between decision times:

$$\log w_{(t+\delta t)-} - \log w_t = \left(\pi_t\mu + (1-\pi_t)r - \tfrac{1}{2}(\pi_t\sigma)^2\right)\delta t + \pi_t\sigma\left(W_{t+\delta t} - W_t\right),$$

At each decision time:

$$w_t = \eta_t \mathbb{1}_{t<t_{\mathrm{RA}}} + (1 - c_t\mathbb{1}_{t\geq t_{\mathrm{RA}}})\left(1 + \frac{p_t}{1-p_t}\,\mathbb{1}_{t>t_{\mathrm{RA}}}\right) w_{t-},$$

where p_t is the probability of dying in year t given survival to $t-1$. idiosyncratic longevity risk is pooled by a tontine.
Outcome:The *replacement ratio* $C_t = c_t/\eta_t$.

Problem: Identifying an individual's preferences is tricky.

Goal Create a tool which allows a pension professional to see sensitivity of the outcome to preference parameters in real time.

The preference model

Let τ be random time of death.

Exponential Kihlstrom–Mirman utility. Summed form:

$$U(C) = \mathbb{E}\left(-\exp\left(-\alpha \sum_{j=t_{\mathrm{RA}}}^{j<\tau} u(C_j)\,\delta t\right)\right), \qquad u(C) = \frac{C^\rho}{\rho} - \frac{a^\rho}{\rho},$$

$\alpha > 0$: *risk aversion* $\quad$ ρ : *satiation* $\quad$ a : *adequacy level*

Recursive form.

$$V_s(C) = \begin{cases} \alpha u(C_s)\,\delta t - \log \mathbb{E}[\exp(-V_{s+1}(C)) \mid \mathcal{F}_s], & s < \tau, \\ 0, & s \geq \tau, \end{cases} \qquad U(C) = -\exp\left(-V_{t_{\mathrm{RA}}}\right).$$

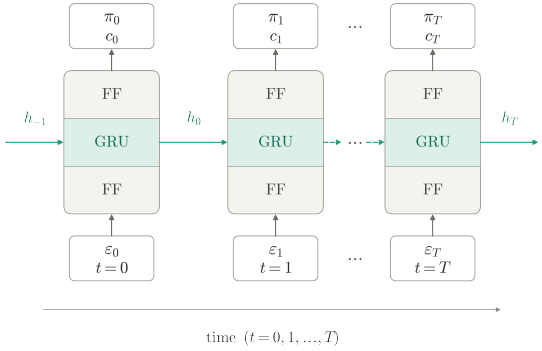
- This is the only Markovian model that can be written both ways. Bommier argues axiomatically that this is the right model under mortality risk.
- Easy to evaluate and understand: a *forward* method, no backward recursion of conditional expectations.

Solving for fixed parameters

Inputs: the Gaussian increments $\epsilon_t = (W_{t+\delta t} - W_t)/\sqrt{\delta t}$ and the time point.
Outputs at each step: π_t^θ and c_t^θ.
Loss evaluated through `logsumexp`.

Network sees only the driving noise and must reconstruct wealth from its own past decisions. Deliberately *not* exploiting Markovianity.

Validation against a provably convergent duality scheme: mean certainty-equivalent error 0.70% (2.03% s.d.) over 100 random parameter combinations.



Standard approach to amortised optimization

The naive method. Sample $\omega^A = (\alpha, \rho, a)$ at random alongside each market scenario ω^M. Minimize

$$\mathbb{E}_{\Omega^A}(\text{Loss}) = \mathbb{E}_{\Omega^A \times \Omega^M}(-\exp(\ldots))$$

Works well when the objective does not vary much over the parameter space e.g. linear-quadratic problems or problems with obvious choice of scale.

Why it fails. Fails when the loss contains an exponential. A small subset of parameter values dominates every gradient step. Fails to learn anything for most parameters.

The SGD error estimate determines the step size

The variance of the gradients sets the step size, and the step size sets the iteration count.

Minimise $L(\omega^M, x) = \frac{1}{N}\sum_i L_i(\omega_i^M, x)$ by $x^{k+1} = x^k - \gamma \nabla L_{i_k}(\omega_{i_k}^M, x^k)$.

Theorem (classical SGD)

For a constant step size $0 < \gamma \leq \frac{1}{4\ell_{\max}}$ and every iteration count $K \geq 1$,

$$\mathbb{E}\left[L(\omega^M, \bar{x}^K) - \inf_x L\right] \leq \frac{\|x^0 - x^*\|^2}{\gamma K} + 2\gamma\sigma_L^*, \qquad \sigma_L^* := \mathbb{V}\left[\nabla L_i(\omega_i^M, x^*)\right].$$

Taking $\gamma = \gamma_0/\sqrt{K}$ makes the bound $O(K^{-1/2})$; it is optimised at

$$\gamma_0 = \frac{\|x^0 - x^*\|}{\sqrt{2\sigma_L^*}} \qquad \Longrightarrow \qquad K \;\geq\; \frac{8\|x^0 - x^*\|^2\,\sigma_L^*}{\epsilon^2}.$$

Key point: Each parameter combination has its own $\sigma_L^*(\omega^A)$, and these differ by orders of magnitude, but there is one training run and so one step size.

Conditionally varying the scale

Work on the product space $\Omega = \Omega^M \times \Omega^A$ with. The parametrised goal is

$$\underset{(c,\pi)\in\mathcal{A}^*}{\operatorname{argmin}} \; -\mathbb{E}\left(U(c) \mid \mathcal{F}^A\right).$$

Conditional expectation is linear. So for any strictly positive $\mathcal{F}^A$-measurable λ,

$$\underset{(c,\pi)\in\mathcal{A}^*}{\operatorname{argmin}} \; -\mathbb{E}\left(\frac{U(c)}{\lambda}\right)$$

has *exactly the same solutions*: at each fixed parameter value the scaling is a positive affine transformation of the objective, so the optimal strategies are untouched. Set $L_{i,\lambda} := L_i/\lambda(\omega_i^A)$.

How this helps. The scale multiplies the gradient, so we can ensure the variance of the gradients and hence the optimal step-size is the same for each parameter combination.

Convergence results

1. The main iteration converges under *any* admissible scale

If (λ^k) is $\mathcal{F}^k$-measurable with $0 < \lambda_{\min} \leq \lambda^k \leq \lambda_{\max} < \infty$ and $\gamma \leq \lambda_{\min}/(4\ell_{\max})$,

$$\mathbb{E}\left[L(\bar{x}_\lambda^K) - \inf_x L\right] \leq \lambda_{\max}\left(\frac{\|x_\lambda^0 - x^*\|^2}{\gamma K} + \frac{2\gamma\sigma_L^*}{\lambda_{\min}^2}\right).$$

The moving scale enters only through $\lambda_{\min}, \lambda_{\max}$: it affects the *speed*, never the *fact*, of convergence.

2. The scale itself can be learned online

Measuring the gradient variance conditioned on the parameter values is itself a form of quadratic optimization problem.

3. Joint convergence

If we bound the scale as in (1) joint convergence of the two optimizations is trivial.

The bottleneck

The theoretically optimal scale is

$$\lambda^*(\omega^A) = \mathrm{SD}\big(\nabla L_i(x^*) \mid \omega^A\big)\,,$$

so estimating it requires the **per-sample gradients** ∇L_i, i.e. one gradient vector for every scenario in the batch.

This grows with the **product** of the batch size and the size of the network.

What this costs. Both methods were given the same 24-hour budget. The gradient-based method was forced onto a batch size of 128 rather than 4096 (32 times smaller) and completed roughly half as many iterations: around *two orders of magnitude less training*.

But we are only tuning the constant in a convergence bound. It is not worth an unbounded computational price.

Scaling using the loss

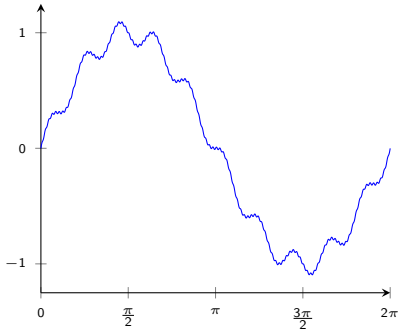
Heuristic scale: $\lambda^*(\omega^A) = \mathrm{SD}\left(L_i(x^*) \mid \omega^A\right)$, the standard deviation of the *loss*.

Fourier coefficients of smooth functions The Fourier C^∞ coefficients of smooth functions decays exponentially. This is called spectral convergence. If the loss is well-scaled, its derivatives will be too.

Pathological functions If derivatives have a larger magnitude than absolute values, this is essentially pathological.

$$\sum_{i=0}^{\infty} \frac{1}{10^i} \sin\left(10^i x\right)$$

Generalizes standard practice This explains the standard practice of scaling data by its standard deviation before fitting a network.



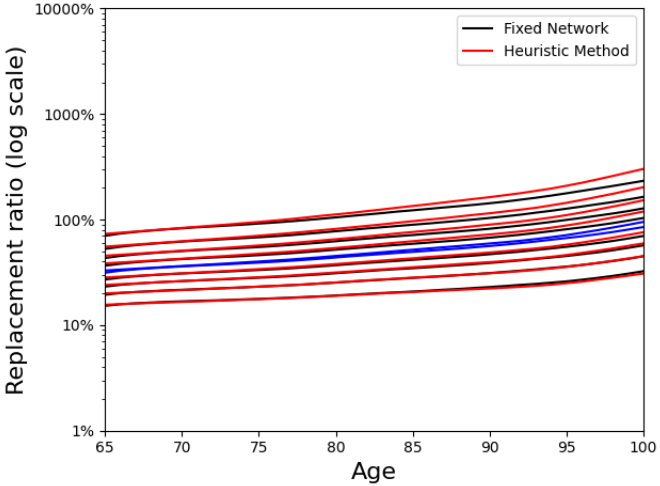
The end result

Certainty-equivalent error against *fixed-parameter* networks, over 100 randomly sampled parameter combinations:

Method	mean error
Gradient-based	4.28%
Heuristic	**0.284%**

and the fixed-parameter networks are themselves within 0.70% of the provably convergent duality scheme.

Both scalings satisfy the hypotheses of the convergence theorem. The theoretically motivated one loses on computation time.



Heuristic method vs. fixed-parameter network