

mcfeedback — Explained

Murray-Claude Feedback Algorithm · Phase 1 · Non-temporal / Spatial-only

1 — What problem is this solving?

Standard neural networks learn via **backpropagation**: a central optimiser computes the exact error contribution of every single weight in the network, then adjusts all of them at once. It works, but it has a fundamental problem — **it is not how brains work**.

A biological synapse has no way to receive a gradient signal from a loss function. It only knows what its immediate neighbours are doing: did the neuron before me fire? Did the neuron after me fire? Is there a reward chemical in my vicinity right now?

The goal of mcfeedback: build a network that learns using only local signals — no synapse ever sees a gradient, knows the global loss, or knows what any non-connected neuron is doing.

2 — The three key ideas

2.1 Eligibility traces (flagging)

Each synapse watches its two neurons and raises a **flag** when they show correlated activity. The flag is a number called the **eligibility trace**. No weight change happens yet — the synapse is only marking itself as a candidate for

reinforcement.

CO-ACTIVATION

pre fires + post fires

→ **positive trace**

These two neurons are working together.

MISMATCH

one fires, the other doesn't

→ **negative trace**

This connection is producing incorrect activations.

CO-SILENCE (ACTIVE AREA)

both silent + high ambient field

→ **positive trace**

Both are suppressed in a busy neighbourhood —
meaningful silence.

IRRELEVANT SILENCE

both silent + quiet area

→ **zero**

Nothing happening here. Not worth flagging.

2.2 Chemical diffusion (the reward gate)

The trace alone does nothing. Weight changes only happen when a **modulatory neuron** releases a chemical signal. These neurons act like the brain's dopamine system: they broadcast "good" or "bad" to everything in their spatial neighbourhood.

After every forward pass, the network compares its output to the target. If the output was correct, modulatory neurons release **positive chemical** (reward). If wrong, **negative chemical** (punishment). The chemical spreads with a spatial falloff — synapses close to a modulatory neuron feel it strongly; distant synapses feel it weakly.

Weight update = $\text{eligibilityTrace} \times \text{chemicalLevel} \times \text{learningRate}$

Positive trace + positive chemical → weight grows (reinforce)

Negative trace + positive chemical → weight shrinks (suppress the mismatch)
Any trace + zero chemical → nothing changes (no reward = no learning)

2.3 Dampening filters

Not every flag deserves equal weight. Three local filters reduce noise:

- **Activity history dampening** — synapses that rarely participate are unreliable. Low history → dampened. Built up over many steps.
- **Information dampening** — a neuron that always fires (or never fires) carries no information. The trace is scaled by $4 \times \text{fireRate} \times (1 - \text{fireRate})$: an inverted-U that peaks at $\text{fireRate} = 0.5$ and approaches zero at the extremes.
- **Ambient relevance dampening** — silence is only meaningful in a busy neighbourhood. A silent neuron surrounded by other silent neurons is irrelevant; surrounded by active ones, its silence is significant.

3 — One training step, in order

- 1 **Clamp inputs.** Input neurons are forced to the values in the current training pattern. They do not compute — they just hold.
- 2 **Forward pass.** Each synapse multiplies its pre-synaptic neuron's output by its weight and adds the result to the post-synaptic accumulator. Then each non-input neuron fires if its accumulator \geq its threshold (binary step function).
- 3 **Ambient field.** Each neuron sums the outputs of its spatial neighbours, weighted by inverse distance. This gives a local "busyness" score used by the co-silence quadrant and dampening.

- 4 **Flagging.** Every synapse inspects its pre and post neuron states and computes a raw eligibility trace using the four-quadrant rule. Activity history is updated here — from the raw trace, before dampening erases it.
- 5 **Dampening.** The raw trace is multiplied by the three dampening factors. Synapses that pass all three filters keep their full trace; others are reduced or zeroed.
- 6 **Reward.** Output neurons are compared to the target. The reward signal maps accuracy to $[-1, +1]$. Modulatory neurons fire based on this signal.
- 7 **Chemical diffusion.** Each modulatory neuron that fired spreads chemical to nearby synapses, falling off with distance. Only synapses within the diffusion radius receive any chemical.
- 8 **Weight update.** `weight += trace × chemical × learningRate`, with the delta clamped and the final weight clamped to $\pm \text{maxWeightMagnitude}$. Weight also decays slightly each step — synapses must be continuously earned.
- 9 **Homeostasis.** Each neuron tracks its long-run fire rate. If it fires too often, its threshold rises; too rarely, it falls. This self-regulation keeps the network from saturating or going silent.

4 — Data layout

The architecture has three tiers. The key design principle: **synapses are primary**. All learning loops iterate the flat synapse array. Neurons are lookup tables that synapses read from.

```
Tier 1 – Neuron position (set once, never changes) id, x, y, z, type, clusterId, neighbourIds[]
Tier 2 – Neuron state (updated every step) output, firedThisCycle, fireRate, threshold,
ambientField Tier 3 – Synapse (the core – all learning lives here) from, to, weight,
eligibilityTrace, activityHistory, chemicalLevel Network neurons: Map<id → Tier1> neuronState:
Map<id → Tier2> synapses: Synapse[] ← flat array, iterated in every step config: Object
```

5 — Experiment 001: pattern association

Two clusters of 30 neurons each. Cluster A receives inputs; Cluster B should produce corresponding outputs. The task: learn 4 binary inversions.

```
Input (Cluster A) → Target (Cluster B) [1, 0, 1, 0, 1] → [0, 1, 0, 1, 0] [1, 1, 0, 0, 0] → [0, 0,
1, 1, 1] [1, 0, 0, 0, 1] → [0, 1, 1, 1, 0] [0, 1, 0, 1, 0] → [1, 0, 1, 0, 1]
```

The same experiment is run four times with different features enabled, to measure each component's contribution in isolation:

Condition	Ambient field	Dampening	Chemical spread
Baseline	off	off	global (uniform)
Ambient only	on	off	global
Dampening only	off	on	global

Full model

on

on

local (spatial falloff)

6 — Bugs found during implementation

Bug 1 — Activity history deadlock

`activityHistory` initialises to 0. `activityHistoryDampening(0)` returns 0. So on step 1, every eligibility trace is multiplied by 0 — all traces become zero. With zero trace, `participated = 0`, so activity history stays 0 forever. Weights never move. The network is permanently stuck from the first step.

Fix

Move `updateActivityHistory` to between flagging and dampening (step 4.5), so it records the raw trace before dampening erases it. Now history builds from the first step and dampening can progressively take effect.

Bug 2 — Linear falloff returns zero for distance > 1

The falloff formula was `Math.max(0, 1 - distance)`. For any synapse more than 1 spatial unit from a modulatory neuron, this returns 0. The "global reward" baseline conditions used radius = 1000 with this formula — so every synapse received zero chemical. The full model's radius = 5 also failed to reach inter-cluster synapses at distance ≈ 11 .

Fix

Normalise: `Math.max(0, 1 - distance / radius)`. Also added a `'constant'` falloff mode (returns 1 regardless of distance) for global-reward conditions.

Bug 3 — Inter-cluster connectivity too sparse for local learning

With `interClusterConnectionProb = 0.1` and 5 input / 5 output neurons, only ~2 direct input→output synapses exist. Three input neurons had zero direct output connections. The signal had to travel 3+ hops through random intermediary neurons. Local learning rules have no mechanism for multi-hop credit assignment without global error signals — which is exactly what this architecture avoids.

Fix

Increased `interClusterConnectionProb` to 0.5, giving ~12 direct input→output connections and reliable 1-hop paths for all input neurons.

7 — Key findings from the first run

Finding 1 — Fire rate runaway without weight decay

The "always fire all output neurons" strategy achieves ~55% accuracy on these patterns — above the 50% baseline for positive reward. This gives a persistent positive reward signal every single step. Weights accumulate until they hit the maximum, neurons fire constantly, homeostatic threshold regulation cannot keep up. The network saturates in a trivial attractor.

Fix: **weight decay** (`weightDecay = 0.005`). Synapses must be continuously reinforced to maintain their strength. Random co-activation alone is no longer enough to sustain high weights.

Finding 2 — Dampening and homeostasis fight each other

The `informationDampening` inverted-U peaks at `fireRate = 0.5`. But `targetFireRate = 0.2` — the homeostatic target. At `fireRate = 0.2`, dampening gives $4 \times 0.2 \times 0.8 = 0.64$: a 36% penalty on every trace, precisely for neurons behaving as intended. The two mechanisms are working against each other. This is a parameter design issue to resolve in Phase 2.

Finding 3 — Ambient field alone is the most effective component

The "Ambient only" condition (ambient field on, no dampening, global reward) achieved and sustained **80% accuracy** from episode 200 onward. The full model (all features enabled) was less stable. This counterintuitive result reveals that the dampening parameters need recalibration before the full model can outperform simpler conditions. The ambient field's co-silence quadrant — rewarding meaningful silence in active neighbourhoods — appears to be the single most valuable addition over baseline Hebbian learning.

8 — What is next (Phase 2)

- Recalibrate `informationDampening` so its peak aligns with `targetFireRate`, not 0.5.
- Investigate the bimodal threshold drift: neurons splitting into always-on and always-off populations instead of settling near the target rate.
- Add temporal structure: refractory periods, spike timing, cycling. The architecture is designed for this but Phase 1 is spatial-only.
- Let modulatory neurons compute their own reward from local inputs, removing the external reward signal.
- Scale up: more clusters, more patterns, larger networks.

mcfeedback Phase 1 — Murray-Claude Feedback Algorithm · Node.js · ES modules · No external ML frameworks · All learning local · No synapse ever sees a gradient.