# Mathematical Workflow of the ML-Prefetcher

Deep Mandal

## 1 What the model is solving

At every L2C access $t$, a small feature vector $\boldsymbol{x}_t \in \mathbb{R}^d$ is constructed (bias, PC bucket, page-offset bucket, last hit/miss, access type, tiny stride memory). A small set of actions $A$ of candidate strides in cache lines is considered (e.g., $A = \{\pm 1, \pm 2, \pm 4, \pm 8\}$).

For each action $a \in A$, weights $\boldsymbol{w}_a \in \mathbb{R}^d$ are maintained and a usefulness probability is computed as

$$p_t(a) = \sigma(\boldsymbol{w}_a^\top \boldsymbol{x}_t) = \frac{1}{1 + e^{-\boldsymbol{w}_a^\top \boldsymbol{x}_t}},$$

interpreted as the probability that a prefetch with stride $a$ will later be useful (hit).

The prefetcher then decides which prefetches to issue (if any), records them as pending, and upon confirmation of usefulness or uselessness, updates the corresponding $\boldsymbol{w}_a$.

This constitutes an online, multi-class classification problem with delayed feedback (a contextual bandit view).

## 2 Why logistic regression (derivation)

The objective is to obtain a calibrated probability $p(y = 1 \mid \boldsymbol{x}, a)$ that a prefetch will be useful. Assume binary labels $y \in \{0, 1\}$ and model

$$\Pr(y = 1 \mid \boldsymbol{x}, a) = \sigma(\boldsymbol{w}_a^\top \boldsymbol{x}).$$

For a labeled pair $(\boldsymbol{x}, y)$, the (regularized) negative log-likelihood is

$$L(\boldsymbol{w}_a) = -\big[y \log \sigma(z) + (1 - y) \log(1 - \sigma(z))\big] + \frac{\lambda}{2} \|\boldsymbol{w}_a\|_2^2, \quad z = \boldsymbol{w}_a^\top \boldsymbol{x}.$$

Gradient:

$$\nabla_{\boldsymbol{w}_a} L = \big(\sigma(z) - y\big)\boldsymbol{x} + \lambda \boldsymbol{w}_a = (\hat{y} - y)\boldsymbol{x} + \lambda \boldsymbol{w}_a.$$

Stochastic gradient descent update with step $\eta$:

$$\boldsymbol{w}_a \leftarrow \boldsymbol{w}_a - \eta\big[(\hat{y} - y)\boldsymbol{x} + \lambda \boldsymbol{w}_a\big] = (1 - \eta\lambda)\, \boldsymbol{w}_a + \eta\, (y - \hat{y})\, \boldsymbol{x}.$$

This is the standard logistic SGD with L2 decay; the factor $(1 - \eta\lambda)$ effects weight decay.

**Why logistic?** Probabilities (0..1) support thresholding and ranking; the loss is convex, enabling stable online learning; computation is $O(d)$ per action.

## 3 Decision rule (why a probability threshold `thr`)

A prefetch has benefit $B$ (expected cycles saved if useful & timely) and cost $C$ (bandwidth, PQ/MSHR pressure, potential pollution). The expected net gain from issuing one prefetch with predicted probability $p$ is

$$\mathbb{E}[\Delta] = p \cdot B - (1 - p) \cdot C.$$

Issue iff $\mathbb{E}[\Delta] \geq 0 \Rightarrow p \geq \dfrac{C}{B+C} \triangleq \texttt{thr}^{\star}$.

Since $B$ and $C$ are typically unknown, a tunable threshold $\texttt{thr} \approx C/(B+C)$ is employed.

- Under tight queue pressure (high $C$), `thr` should be increased.
- When memory is underutilized (low $C$), `thr` can be decreased.

This yields the decision-theoretic basis for `thr`.

# 4 Why a cap max_out

Multiple candidates may be prefetched per access, but queues and bandwidth are finite. If the L2 prefetch queue has free capacity $Q_{\text{free}}$ and a safety margin $m$ is desired, then a conservative per-access cap is

$$\texttt{max\_out} \leq Q_{\text{free}} - m.$$

Another view: memory latency $L$ (cycles) and desired lookahead distance $D$ (lines) together determine the effective degree; speculating beyond this provides little timeliness benefit. `max_out` limits speculation to what the system can absorb.

# 5 Why a TIMEOUT

A prefetch is considered "useful" only if it is hit by a subsequent demand before eviction or staleness. If it is never used within a horizon, it is labeled useless. Timeliness is approximated with a fixed age horizon:

$$\text{if age} \geq \texttt{TIMEOUT} \;\Rightarrow\; y = 0.$$

`TIMEOUT` is specified in L2C accesses rather than cycles so that it is tied to the stream hitting L2. A practical heuristic is

$$\texttt{TIMEOUT} \approx \frac{\text{miss penalty (cycles)}}{\text{avg L2C accesses per cycle}} \times \text{fudge factor}.$$

Purpose: generate negative labels without indefinite waiting and penalize late/never-used prefetches (implicitly teaching timeliness).

# 6 Why L2 weight decay $\lambda$ (and its "half-life")

Each update multiplies the weights by $(1 - \eta\lambda)$. After $k$ labeled updates with no signal, a weight decays to

$$\|\boldsymbol{w}\| \approx (1 - \eta\lambda)^k \|\boldsymbol{w}_0\| \approx e^{-\eta\lambda k} \|\boldsymbol{w}_0\|.$$

Defining a half-life in updates $k_{1/2}$ via $e^{-\eta\lambda k_{1/2}} = \frac{1}{2}$ gives

$$k_{1/2} = \frac{\ln 2}{\eta\lambda}.$$

Thus, $\lambda$ can be chosen to match the desired forgetting rate for stale phases. Example: with $\eta = 10^{-3}$ and target half-life $k_{1/2} = 700$ updates, $\lambda \approx \dfrac{\ln 2}{700 \cdot 10^{-3}} \approx 0.99 \times 10^{-3}$.

## 7 How to pick a learning rate $\eta$ (stability)

The logistic loss has Lipschitz gradient $L \le \frac{1}{4}\|\boldsymbol{x}\|^2 + \lambda$ (since $\sigma'(z) \le 1/4$). With $\|\boldsymbol{x}\|^2 \le R^2$ (tiny one-hots $\to$ small $R^2$), a conservative condition is

$$\eta \lesssim \frac{1}{\frac{1}{4}R^2 + \lambda}.$$

In practice, with small one-hot features and modest $d$, $\eta \in [10^{-3}, 10^{-2}]$ is typically stable. If oscillations in accuracy are observed, a smaller $\eta$ is advisable; if convergence is sluggish, a slightly larger $\eta$ may help.

## 8 Why $\varepsilon$-greedy exploration and the schedule $\varepsilon_t$

Early in training, the model's probabilities are uncalibrated. Pure exploitation of $\max_a p_t(a)$ can cause stagnation. $\varepsilon$-greedy exploration occasionally tries plausible alternatives (e.g., swapping $+1$ with $-1$) to discover better options.

A simple linear decay is

$$\varepsilon_t = \varepsilon_{\text{start}} + (\varepsilon_{\text{end}} - \varepsilon_{\text{start}}) \cdot \min\left(1, \frac{t}{T_{\text{decay}}}\right).$$

A high initial value promotes exploration, tapering to a small floor (e.g., 0.01). $T_{\text{decay}}$ controls the duration of active exploration. An exponential decay $\varepsilon_t = \varepsilon_0 \alpha^t$ is also common. From a bandit perspective, exploration ensures regret shrinks over time.

## 9 Windowed controller (why $W$, and how the rules arise)

The aim is to achieve high accuracy and sufficient coverage without exceeding bandwidth constraints. Consider the constrained objective:

$$\max\ \mathbb{E}[\text{useful}] \quad \text{s.t.} \quad \mathbb{E}[\text{issued}] \le \text{budget}.$$

The Lagrangian $L = \mathbb{E}[\text{useful}] - \mu\,\mathbb{E}[\text{issued}]$ again implies a threshold rule $p \ge$ a $\mu$-dependent bound (as in §3).

Because $\mu$ is unknown, `thr` and `max_out` are adapted from windowed statistics. Over the last $W$ L2C accesses, measure: Issued $I$, Useful $U$, Demand misses $M$.

Accuracy $\alpha = U/\max(1, I)$.

(With-run) coverage $\kappa \approx U/\max(1, M)$.

**Rules (heuristic control).**

- If $\kappa < \kappa_{\min}$ and $\alpha \ge \alpha_{\text{hi}}$, the controller may lower `thr` slightly and increase `max_out` by 1.
- If $\alpha < \alpha_{\text{lo}}$, `thr` is increased and `max_out` is set to 1.

**Why a window $W$?** It trades variance against responsiveness. The standard error of $\alpha$ over the window is $\approx \alpha(1-\alpha)/I$. When hundreds or thousands are issued per window, estimates are stable. Typical $W$: 2–8K L2C accesses.

## 10 Putting it all together (the full workflow)

**Initialization:** weights are zeroed ($\boldsymbol{w}_a = \mathbf{0}$); `thr`, `max_out`, `TIMEOUT`, $\eta$, $\lambda$, $W$, and the $\varepsilon$ schedule are set; pending structures and window counters are cleared.

**On every L2C access:**

1. Construct features $\boldsymbol{x}_t$ (tiny one-hots + flags).
2. For each stride $a$, compute $p_t(a) = \sigma(\boldsymbol{w}_a^\top \boldsymbol{x}_t)$.
3. Rank actions by $p_t(\cdot)$; apply an $\varepsilon_t$-greedy swap with probability $\varepsilon_t$.
4. Issue the top actions with $p_t(a) \geq$ `thr`, up to `max_out`, subject to back-pressure (PQ/MSHR).
5. For each issued line $L_{\text{pf}}$, record pending$[L_{\text{pf}}] = (a, \boldsymbol{x}_t, \text{age} = 0)$ and increment the window Issued counter.

**Aging & matching:**

- When a demand hits a pending line $L_{\text{pf}}$, extract $(a, \boldsymbol{x}_{\text{issue}})$, set $y = 1$, perform the SGD update, increment Useful, and erase the pending entry.
- Pending entries are periodically aged; if age $\geq$ `TIMEOUT`, set $y = 0$, perform the SGD update, and erase the entry.

**Learning update (for each labeled $(a, \boldsymbol{x}, y)$):**

$$\hat{y} = \sigma(\boldsymbol{w}_a^\top \boldsymbol{x}), \quad \boldsymbol{w}_a \leftarrow (1 - \eta\lambda)\boldsymbol{w}_a + \eta\,(y - \hat{y})\,\boldsymbol{x}.$$

**Every $W$ L2C accesses:**

1. Compute $\alpha = U/\max(1, I)$ and $\kappa \approx U/\max(1, M)$.
2. Adapt `thr` and `max_out` according to the controller rules.
3. Reset the window counters.

**Report:** include IPC (from core counters), Accuracy $= U/I$, Coverage (true) $= U/M_{\text{baseline}}$ (requiring a separate no-prefetch baseline run), counts of issued/useful/useless prefetches, and indicators of queue pressure.

## 11 Practical "derivation $\rightarrow$ parameter" cheatsheet

**`thr` (probability threshold).** Derived from $pB - (1-p)C \geq 0 \Rightarrow p \geq C/(B+C)$. Tuned as a proxy for this bound; increase under pressure, decrease to improve coverage.

**`max_out` (cap per access).** Determined by queue/bandwidth constraints; it should remain below PQ/MSHR free capacity and below the useful lookahead required to hide latency.

**`TIMEOUT`.** Approximates a timeliness deadline; converts missing/late prefetches into negative labels.

**$\eta$ (learning rate).** From logistic smoothness: $\eta \lesssim 1/(\frac{1}{4}\|\boldsymbol{x}\|^2 + \lambda)$. Typical values $10^{-3}$–$10^{-2}$; reduce if unstable.

**$\lambda$ (L2 decay).** Controls forgetting; half-life $k_{1/2} = (\ln 2)/(\eta\lambda)$. The horizon $k_{1/2}$ is specified in labeled updates rather than cycles.

**$W$ (window).** Controls noise of $\alpha, \kappa$. Larger $W$ improves stability; smaller $W$ increases responsiveness.

**$\varepsilon_t$ (exploration).** Bandit-driven; linear or exponential decay schedules are common. A small floor is maintained for adaptability.

## 12 Why this gives good behavior

Calibrated probabilities enable natural thresholding. The cost/benefit derivation yields principled aggressiveness (`thr`). Pending+`TIMEOUT` converts delayed feedback into supervised signals, penalizing late/unused lines. L2 decay provides smooth forgetting of stale patterns and phase changes. $\varepsilon$-greedy exploration avoids stagnation and adapts to shifts. The windowed controller steers the accuracy–coverage trade-off within bandwidth limits to preserve IPC gains.