# TRANSFORMERS LEARN LATENT MIXTURE MODELS IN-CONTEXT VIA MIRROR DESCENT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Sequence modelling requires determining which past tokens are causally relevant from the context and their importance: a process inherent to the attention layers in transformers, yet whose underlying learned mechanisms remain poorly understood. In this work, we formalize the task of estimating token importance as an in-context learning problem by introducing a novel framework based on Mixture of Transition Distributions, whereby a latent variable, whose distribution is parameterized by a set of unobserved mixture weights, determines the influence of past tokens on the next. To correctly predict the next token, transformers need to learn the mixture weights in-context. We demonstrate that transformers can implement Mirror Descent to learn the mixture weights from the context. To this end, we give an explicit construction of a three-layer transformer that exactly implements one step of Mirror Descent and prove that the resulting estimator is a first-order approximation of the Bayes-optimal predictor. Corroborating our construction and its learnability via gradient descent, we empirically show that transformers trained from scratch converge to this solution: attention maps match our construction, and deeper models' performance aligns with multi-step Mirror Descent.

## 1 INTRODUCTION

In recent years, Machine Learning has been transformed by large language models (LLMs). These massive, complex models achieve unprecedented performance across diverse tasks beyond text generation (Bubeck et al., 2023). A striking example is in-context learning (ICL) (Brown et al., 2020; Min et al., 2022), where models adapt to new tasks using only examples in the prompt without parameter updates. Mechanistic interpretability has made significant strides in explaining this phenomenon, revealing that transformers can implement computational circuits (Elhage et al., 2021; Olsson et al., 2022; D'Angelo et al., 2025) that mimic known algorithms. For instance, in settings like linear regression, they learn to implement gradient-based optimization (Garg et al., 2022; Akyürek et al., 2022; Von Oswald et al., 2023a;b; Zhang et al., 2023; Ahn et al., 2023; Mahankali et al., 2024), while for Markov Chains, they implement counting-based estimators for the transition probabilities (Nichani et al., 2024; Edelman et al., 2024; Bietti et al., 2023a; Rajaraman et al., 2024; Ildiz et al., 2024; Svete & Cotterell, 2024; Chen et al., 2024). However, these successes are confined to problems where all the sequences rely on the same, fixed causal structure: the relationship between tokens is static. For instance, in the case of regression, the model only needs to learn that every even token depends on the previous odd token; in Markov chains, it learns that the next token depends only on the previous one.

Real-world sequential data, particularly language, defies such simplicity. The meaning of a sentence arises not from the fixed sequence of word meanings, but from the dynamic causal links between the words that must be inferred from the context. These underlying structures, which are fundamental to language, are latent variables hidden from direct observation. Consider the sentence in Figure 1. To predict the final action, a model cannot rely on simple recency. It must infer a latent structure: that "dog" is the agent, "ball" is the relevant object, and "bird" is a distractor. The influence of a past token is not merely a function of its position, but of its inferred role. This ability to infer and reason over unobserved variables, be it syntactic roles, speaker intent, or the causal topic of a discourse, is a hallmark of intelligence. A robust model must therefore move beyond fixed dependencies and learn to infer this latent structure, dynamically identifying which tokens are causally relevant in a given context. These considerations give rise to the core question of this paper:

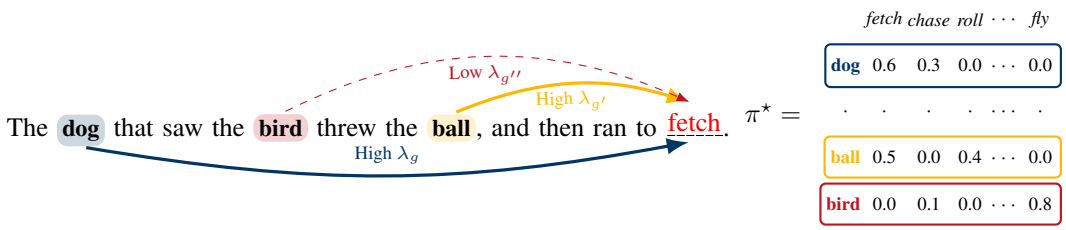


Figure 1: To predict the final word, the model must infer the causal relevance of past tokens. Our MTD framework models this by separating a static, context-free unigram ($\pi^\star$) from dynamic, context-dependent weights ($\boldsymbol{\lambda}$) that are inferred in-context. The model learns to assign high weights to the causally relevant positions ('dog') and ('ball'), activating their respective slices of the unigram (e.g., $\pi^\star(\text{dog}, \cdot)$ favouring verbs like *fetch* or *chase*). Conversely, a low weight is assigned to the distractor ('bird'), suppressing its influence.

*Can transformers infer latent structures in-context, and what algorithm do they learn?*

To investigate this question, we introduce a synthetic task based on the Mixture of Transition Distributions (MTD) model (Raftery, 1985; André Berchtold, 2002). This framework recasts token-importance estimation as learning latent mixture weights in-context. While prior work have used mixture models in regressions (Pathak et al., 2024) or HMMs (Xie et al., 2022) to study ICL, they did not unveil the mechanism through which transformers learn to infer the latent mixture weights.

We make the following contributions:
(I) A framework for ICL based on latent variables using MTD: we create a synthetic task that frames the estimation of the influence of past tokens as learning latent mixture weights in-context. (II) We identify Mirror Descent as the algorithm transformers can implement to solve this task and provide a construction of a 3-layer transformer that exactly implements one-step of this algorithm. (III) We empirically demonstrate that transformers trained with Adam actually learn this estimator and the learned attention patterns math our construction and that deeper models match multi-step MD. (IV) We prove that this estimator is an approximation to the Bayes-optimal solution. Taken together, our results extend the gradient-based interpretation of ICL beyond the regression setting. We show for the first time that the same algorithmic perspective holds also in sequential domains over finite sets of discrete tokens. Our results reveal that Transformers effectively can implement mirror descent to in-context learn latent parameters of Markov-chain tasks, providing a gradient-based explanation of ICL in sequence modeling. We defer to the Appendix C a more detailed discussion of related work.

**Why MTD?** Our choice of the MTD model is motivated by the desire to capture both in-weight and in-context learning in a single, controlled setting. In contrast to much of the ICL literature, where tasks are designed so that nearly all useful structure must be inferred from the prompt (e.g., gradient-descent ICL in linear regression or counting estimators for simple Markov chains), our setup explicitly assumes that some statistical structure, namely the transition matrix $\pi^\star$, can be stored in the model's weights during pretraining and reused at inference time, while the mixture weights $\boldsymbol{\lambda}$ are inferred and adapted in-context from a single sequence. This mirrors a plausible regime for large language models, which cannot memorize full high-order $n$-grams due to their sample complexity growing exponentially with $n$ but can realistically encode lower-order $n$-grams in their weights and dynamically reweight different lags depending on the given prompt. To the best of our knowledge, this is the first in-context learning framework that explicitly couples an in-weight component (learning $\pi^\star$) with an in-context component (inferring $\boldsymbol{\lambda}$), providing a natural testbed for studying the fundamental in-weight / in-context dichotomy in LLMs.

## 2 Disentangled Transformers

The *disentangled transformer* (Friedman et al., 2023) is a modification of a standard Transformer with relative positional encodings (RPE) (Shaw et al., 2018), designed for enhanced interpretability by: (1) removing MLPs; (2) replacing residual connections with concatenation, creating an explicit residual stream that preserves computations from all previous layers; and (3) simplifying the attention mechanism to use a single attention matrix instead of query and key, and incorporates the value in the output matrix. Nichani et al. (2024) demonstrated that disentangled trans-

formers are equivalent to standard transformers using only attention layers. The model maps a sequence of tokens $\boldsymbol{s} = (s_1, \ldots, s_T)$ from a finite alphabet $\mathcal{S}$ to a sequence of vectors. Each token $s_i$ is represented by its one-hot vector $\boldsymbol{e}_{s_i} \in \{0,1\}^{|\mathcal{S}|}$. This yields the initial representations $\boldsymbol{H}^{(0)} = (\boldsymbol{e}_{s_1}, \ldots, \boldsymbol{e}_{s_T}) \in \mathbb{R}^{d_0 \times T}$, whith $d_0 = |\mathcal{S}|$. The model consists of $L$ layers. Let $\boldsymbol{H}^{(l-1)} \in \mathbb{R}^{d_{l-1} \times T}$ be the input to layer $l$, where $\boldsymbol{h}_i^{(l-1)}$ is its $i$-th column vector. For each head $h \in \{1, \ldots, H_l\}$, the pre-softmax attention score $e_{ij}^{(l,h)}$ from token $i$ to $j$ is computed as:

$$e_{ij}^{(l,h)} = (\boldsymbol{h}_i^{(l-1)})^\top \boldsymbol{W}_A^{(l,h)} \boldsymbol{h}_j^{(l-1)} + (\boldsymbol{h}_i^{(l-1)})^\top \boldsymbol{r}_{ij}^{(l,h)}. \tag{1}$$

Here, $\boldsymbol{W}_A^{(l,h)} \in \mathbb{R}^{d_{l-1} \times d_{l-1}}$ is a learnable attention matrix and $\boldsymbol{r}_{ij}^{(l,h)} = (\boldsymbol{R}_A^{(l,h)})_{i-j+1,:}$ is a relative positional encoding vector retrieved from a learnable lookup table $\boldsymbol{R}_A^{(l,h)} \in \mathbb{R}^{(T) \times d_{l-1}}$. The attention weights are computed via a causally masked softmax: $\mathcal{A}_{ij}^{(l,h)} = [\text{softmax}(\boldsymbol{e}_i^{(l,h)})]_j$, where $\boldsymbol{e}_i^{(l,h)}$ is the vector of scores. The output of a single attention head is formed by concatenating the content vector $\boldsymbol{h}_j^{(l-1)}$ with a positional value embedding $\boldsymbol{r'}_{ij}^{(l,h)}$ before the sum and it is then concatenated with the input to form the layer output:

$$\hat{\boldsymbol{h}}_i^{(l,h)} = \sum\nolimits_{j=1}^{T} \mathcal{A}_{ij}^{(l,h)} \,\text{Concat}\left(\boldsymbol{h}_j^{(l-1)}, \boldsymbol{r'}_{ij}^{(l,h)}\right) \quad \boldsymbol{H}^{(l)} = \text{Concat}\left(\boldsymbol{H}^{(l-1)}, \hat{\boldsymbol{H}}^{(l,1)}, \ldots, \hat{\boldsymbol{H}}^{(l,H_l)}\right).$$

The positional value embeddings $\boldsymbol{r'}_{ij}^{(l,h)}$ are retrieved from a second lookup table $\boldsymbol{R}_V^{(l,h)} \in \mathbb{R}^{T \times d_R}$, where $d_R$ is a fixed hyperparameter. The dimension of the representation thus grows according to the recurrence $d_l = d_{l-1} + H_l \cdot (d_{l-1} + d_R)$. After $L$ layers, a final linear layer with matrix $\boldsymbol{W}_O \in \mathbb{R}^{|\mathcal{S}| \times d_L}$ maps the final representation $\boldsymbol{H}^{(L)}$ to logit predictions.

## 3 Mixture of Transition Distributions for In-Context Learning

The Mixture of Transition Distributions (MTD) model, introduced by Raftery (1985), is a higher-order Markov chain that offers a parsimonious representation of long-range dependencies. The core idea is to model the probability of the next state as a convex combination of several first-order transition probabilities, each conditioned on a different past state (or lag).

**Model Definition:** Let $\boldsymbol{Y} = (Y_1, \ldots, Y_T)$ be a sequence of random variables taking values in a finite alphabet $\mathcal{Y} = \{1, \ldots, q\}$. The MTD model of order $m$ explains this sequence by positing a corresponding sequence of unobserved latent variables $\boldsymbol{Z} = (Z_{m+1}, \ldots, Z_T)$, where each $Z_t \in \{1, \ldots, m\}$ acts as a switch, selecting which of the $m$ previous tokens, $Y_{t-1}, \ldots, Y_{t-m}$, will influence the current token $Y_t$. The selection of this lag is a random event, governed by the mixture weights $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ with $\lambda_g \geq 0$ for all $g$ and $\sum_{g=1}^{m} \lambda_g = 1$, such that the probability of choosing lag $g$ is given by $\mathbb{P}(Z_t = g) = \lambda_g$. Once the lag $Z_t = g$ is sampled, the next token $Y_t$ is generated from a first-order transition that depends only on the state at the sampled position, $Y_{t-g}$. This is captured by a transition matrix $\boldsymbol{\pi} \in \mathcal{P}$ with $\mathcal{P}$ the set of $q \times q$ row-stochastic matrix defining the conditional probabilities $\pi(i,j) = \mathbb{P}(Y_t = j \mid Y_{t-g} = i, Z_t = g)$. By marginalizing over the unobserved latent variable $Z_t$, we obtain the model's overall predictive distribution:

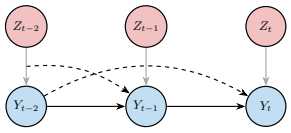


Figure 2: MTD for m=2.

**Definition 1** (Mixture Transition Distribution)**.** *A sequence of random variables $\boldsymbol{Y}$ follows an $m$-th order MTD model if for all $t > m$ and any history $\boldsymbol{y}_1^{t-1}$, the conditional probability of $Y_t$ is:*

$$\mathbb{P}(Y_t = y_t \mid \boldsymbol{Y}_1^{t-1} = \boldsymbol{y}_1^{t-1}, \boldsymbol{\lambda}) = \sum\nolimits_{g=1}^{m} \lambda_g \, \pi(y_{t-g}, y_t), \tag{2}$$

*where the parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ are the mixture weights and $\boldsymbol{\pi}$ the transition matrix.*

This structure allows to model different effective contexts, making it more flexible than a first-order Markov chain preserving tractability by requiring only $m - 1 + q(q-1)$ parameters compared to the $q^m(q-1)$ of a full $m$-th order Markov chain.

**The In-Context Learning Task:** We design an in-context learning task based on the MTD model structured as follows: Given a transition matrix $\boldsymbol{\pi}$ we generate sequences $\boldsymbol{y}$, by sampling for each a

new vector of mixture weights $\boldsymbol{\lambda}$ from a prior $\boldsymbol{\lambda} \sim \text{Dirichlet}(\boldsymbol{\alpha} = \mathbf{1})$. This $\boldsymbol{\lambda}$ vector is the hidden *task* that defines the statistical structure of that particular sequence. The sequence $\boldsymbol{y} = (y_1, \ldots, y_T)$ is then generated according to the MTD model in equation 2. The learning objective for a model $f \in \mathcal{F}$, such as a transformer, is to predict the next token $y_t$ given the context $\boldsymbol{y}_1^{t-1}$:

$$\inf_{f \in \mathcal{F}} \mathbb{E}_{\boldsymbol{\lambda} \sim \text{Dirichlet}(\boldsymbol{\alpha})} \mathbb{E}_{\boldsymbol{y} \sim \text{MTD}(\boldsymbol{\lambda}, \boldsymbol{\pi}^\star)} \left[ \text{KL} \left( p(Y_t | \boldsymbol{y}_1^{t-1}, \boldsymbol{\lambda}) \,\|\, f(\boldsymbol{y}_1^{t-1}) \right) \right] . \tag{3}$$

To perform this prediction optimally, the model must effectively learn the mixture weights $\boldsymbol{\lambda}$ of the latent variables $Z_t$, which are not directly observable in the sequence and differ across sequences:

***In-Context Task**: Learn the unknown mixture weights $\boldsymbol{\lambda}$ given a single sequence $\boldsymbol{y}$.*

**The Bayes Optimal Solution:** The solution to the ICL task (Eq. 3) is the Bayesian predictive distribution, $p(Y_{t+1} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})$.

**Proposition 1** (Bayes-Optimal MTD Predictor)**.** *Given the MTD model with a known transition matrix $\boldsymbol{\pi}$, a prior $p(\boldsymbol{\lambda}|\boldsymbol{\alpha})$, and an observed sequence $\boldsymbol{y}_1^t$, the Bayesian predictive distribution for $Y_{t+1} = j$ is a convex combination of the lag-specific transition probabilities, weighted by the posterior mean of the mixture weights $\boldsymbol{\lambda}$:*

$$p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) = \sum_{g=1}^{m} \hat{\lambda}_g^{\textit{Bayes}} \cdot \pi(y_{t+1-g}, j) \quad \hat{\lambda}_g^{\textit{Bayes}} := \mathbb{E}[\lambda_g \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}] = \int_{\Delta_{m-1}} \lambda_g \cdot p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) \, d\boldsymbol{\lambda}, \tag{4}$$

*where $p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) \propto p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda}) p(\boldsymbol{\lambda} \mid \boldsymbol{\alpha})$ is the posterior distribution with $p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda}) = \prod_{k=m+1}^{t} \left( \sum_{h=1}^{m} \lambda_h \pi(y_{k-h}, y_k) \right)$ being the likelihood.*

The derivation is provided in Appendix E. While elegant, the Bayes-optimal predictor is analytically intractable. The core issue is that the Dirichlet prior is not conjugate to the MTD likelihood, meaning the posterior distribution does not have a closed form. Consequently, the integral defining the posterior mean $\hat{\lambda}_g^{\text{Bayes}}$ cannot be computed directly, necessitating the use of approximation methods.

**Mirror Descent (MD):** The intractability of the posterior mean motivates considering simpler point estimates, such as the Maximum Likelihood Estimate (MLE) or the Maximum A Posteriori (MAP) estimate which are equivalent under uniform Dirichlet prior. However, analytical computation of either the MLE or MAP is intractable necessitating iterative optimization methods. Methods such as Expectation–Maximization (EM) or Mirror Descent (MD) are preferred over standard gradient descent, as they are naturally adapted to the geometry of the simplex. For an extended discussion, see Appendix G. Furthermore, even if the MAP estimate could be found, it represents the mode of the posterior distribution, which does not coincide with the posterior mean in the Bayes-optimal predictor therefore leading to suboptimal predictions.

Mirror Descent, is a first-order method particularly well-suited for optimizing over the probability simplex $\Delta_{m-1}$ Nemirovskij & Yudin (1983); Beck & Teboulle (2003). By using a Bregman divergence based on the negative entropy potential, MD results in the **Exponentiated Gradient (EG)** algorithm, which has a simple multiplicative update rule (see Appendix G.4 for details):

$$\lambda_g^{(k+1)} = \frac{\lambda_g^{(k)} \exp(\eta \cdot \nabla_\lambda \ell(\boldsymbol{\lambda}^{(k)})_g)}{\sum_{h=1}^{m} \lambda_h^{(k)} \exp(\eta \cdot \nabla_\lambda \ell(\boldsymbol{\lambda}^{(k)})_h)}, \tag{5}$$

where $\eta > 0$ is the learning rate and $\nabla_\lambda \ell(\boldsymbol{\lambda}^{(k)})$ the gradient of the log-likelihood evaluated at $\boldsymbol{\lambda}^{(k)}$. Instead of iterating the EG algorithm to convergence, we analyze a non-iterative estimator derived from a single update step, initialized at the center of the probability simplex ($\boldsymbol{\lambda}^{(0)} = (1/m, \ldots, 1/m)$). This approach yields a computationally efficient, regularized approximation of the MLE, which we demonstrate serves as an effective proxy for the posterior mean.

**Proposition 2** (One-Step MD Estimator)**.** *Initializing the EG algorithm (Eq. 5) at $\boldsymbol{\lambda}^{(0)} = (1/m, \ldots, 1/m)$ and applying a single update step for the MTD model yields the estimator:*

$$\hat{\lambda}_g^{\textit{MD}} := \frac{\exp\left(\eta \cdot m \sum_{k=m+1}^{t} \gamma_k(g)\right)}{\sum_{j=1}^{m} \exp\left(\eta \cdot m \sum_{k=m+1}^{t} \gamma_k(j)\right)} \quad \gamma_k(g) := p(Z_k = g \mid \boldsymbol{y}_1^k, \boldsymbol{\lambda}^{(0)}) = \frac{\pi(y_{k-g}, y_k)}{\sum_{h=1}^{m} \pi(y_{k-h}, y_k)}, \tag{6}$$

*where $\gamma_k(g)$ is the posterior responsibility of lag $g$ at step $k > m$, under the uniform prior.*

The derivation is provided in Appendix G.4. This one-step estimator computes, for each step $k$ in the sequence, the posterior probability that lag $g$ was responsible for generating $y_k$ (assuming all lags equally likely a priori). These responsibilities are summed across the sequence and fed into a softmax function to produce the estimate $\hat{\boldsymbol{\lambda}}^{\text{MD}}$ with the learning rate $\eta$ controlling its sharpness.

## 4 Transformers implement One-Step of Mirror Descent

We present our main theoretical result: a constructive proof that the one-step MD estimator can be implemented by a Transformer. The crucial mechanism relies on relative position encodings to correctly route information, allowing the self-attention layer to compute the posterior responsibilities.

**Proposition 3** (Transformer Implementation of the One-Step MD Estimator)**.** *Given an MTD model of order $m$ with a known transition matrix $\boldsymbol{\pi}^\star \in \mathcal{P}$. For any sequence $\boldsymbol{y}_{1:T}$ of length $T \geq m$, there exists a three-layer disentangled Transformer $\tilde{\mathcal{T}}$ with single head, $d_0 = q$ and $d_R \geq m$ that implements the one-step MD estimator. The Transformer produces the predictive distribution for the next token $Y_{t+1}$ at position $t$ as:*

$$\tilde{\mathcal{T}}(\boldsymbol{y}_{1:T})_T = \sum_{g=1}^{m} \tilde{\lambda}_g(\boldsymbol{y}_{1:T}) \cdot \boldsymbol{\pi}(y_{T+1-g}, :) \qquad \tilde{\lambda}_g(\boldsymbol{y}_{1:T}) = \frac{\exp\left(\frac{\beta}{T-m} \sum_{i=m+1}^{T} \gamma_i(g)\right)}{\sum_{h=1}^{m} \exp\left(\frac{\beta}{T-m} \sum_{i=m+1}^{T} \gamma_i(h)\right)}, \tag{7}$$

*with the weights $\tilde{\boldsymbol{\lambda}}(\boldsymbol{y}_{1:t})$ computed exactly as the one-step MD estimate and $\beta$ is a learnable parameter corresponding to the scaled learning rate of the MD algorithm.*

In the following we prove Proposition 3 by explicitly constructing a 3-layer disentangled Transformer that implements the one-step MD estimator. The first layer computes the posterior responsibilities $\gamma_i(g)$, the second layer computes the logits $\sum_{i=m+1}^{T} \gamma_i(g)$, and the third layer produces the final estimate vector $\tilde{\boldsymbol{\lambda}}(\boldsymbol{y}_{1:T})$ within its attention weights.

**Layer 1, Posterior Responsibilities:** The first layer uses the atteniton matrix $\boldsymbol{W}_A^{(1)}$ to compute the posterior responsabilities and the relative positional encoding $\boldsymbol{r}'_{ij}$ to store it in the residual stream:

$$\boldsymbol{W}_A^{(1)} = \log \boldsymbol{\pi}^\star, \; (\boldsymbol{R}_A^{(1)})_{k,:} = \begin{cases} +\delta_1 \cdot \mathbf{1}^\top & 2 \leq k \leq m+1 \\ -\delta_1 \cdot \mathbf{1}^\top & \text{otherwise} \end{cases}, \; (\boldsymbol{R}_V^{(1)})_{k,:} = \begin{cases} \boldsymbol{e}_{k-1}^\top & 2 \leq k \leq m+1 \\ \mathbf{0}^\top & \text{otherwise} \end{cases}$$

$$\boldsymbol{R}_A^{(1)\top} = \delta_1 \left(\begin{array}{c|ccc|ccc} \text{1} & \text{2} & \dots & \text{m+1} & \text{m+2} & \dots & \text{T} \\ \hline -1 & 1 & \dots & 1 & -1 & \dots & -1 \\ -1 & 1 & \dots & 1 & -1 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ -1 & 1 & \dots & 1 & -1 & \dots & -1 \end{array}\right) \qquad \boldsymbol{R}_V^{(1)\top} = \left(\begin{array}{c|ccc|ccc} \text{1} & \text{2} & \dots & \text{m+1} & \text{m+2} & \dots & \text{T} \\ \hline 0 & 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{array}\right)$$

where $\log$ is applied element-wise, $k = i - j + 1$ is the relative position between token $i$ and $j$ shifted by 1. The lookup table $\boldsymbol{R}_A^{(1)}$ biases attention to focus only on the first $m$ relative positions (the lags). The lookup table $\boldsymbol{R}_V^{(1)}$ uses one-hot vectors $\boldsymbol{e}_k$ to copy the computed attention weight for a specific lag into the corresponding dimension of the output vector, thereby storing the responsibilities for different lags in distinct positions. The attention score $e_{ij}$ for this layer is computed as per Equation 1. Given that the input is one-hot encoded, i.e., $\boldsymbol{h}_i^{(0)} = \boldsymbol{e}_{y_i}$, the score becomes:

$$e_{ij} = \boldsymbol{e}_{y_i}^\top (\log \boldsymbol{\pi}^\star) \boldsymbol{e}_{y_j} + \boldsymbol{e}_{y_i}^\top \boldsymbol{r}_{ij}^{(1)} = \log \pi^\star(y_i, y_j) + \begin{cases} +\delta & \text{if } 1 \leq i - j \leq m \\ -\delta & \text{otherwise} \end{cases},$$

where we used that the RPE vector $\boldsymbol{r}_{ij}$ is constant with respect to the token values $y_i$. For a large $\delta_1$, the softmax only attends to keys $j$ such that their relative position $k = i - j$ is within the range $[1, m]$. The causal attention weights $\mathcal{A}_{ij}^{(1)}$ for $j \leq i$ are given by: $\mathcal{A}_{ij}^{(1)} = \frac{\exp(e_{ij})}{\sum_{j'=1}^{i} \exp(e_{ij'})} = \frac{\pi^\star(y_i, y_j) \exp(\boldsymbol{e}_{y_i}^\top \boldsymbol{r}_{ij}^{(1)})}{\sum_{j'=1}^{i} \pi^\star(y_i, y_{j'}) \exp(\boldsymbol{e}_{y_i}^\top \boldsymbol{r}_{ij'}^{(1)})}$. In the limit $\delta_1 \to \infty$, the behavior of the softmax changes based on the

position $i$. For the first token ($i = 1$), the only valid key is itself ($j' = 1$), which means the attention is entirely self-contained, resulting in $\mathcal{A}^{(1)}_{11} = 1$. For any subsequent token ($i > 1$), the terms in the denominator corresponding to lags $k \in [1, m]$ are scaled by $\exp(\delta_1)$, while all other terms, including the diagonal ($k = 0$), are scaled by $\exp(-\delta_1)$ and vanish. This yields the following limit for $i > 1$:

$$\mathcal{A}^{(1)}_{ij} = \begin{cases} \frac{\pi^\star(y_i, y_j)}{\sum_{k=1}^{\hat{m}} \pi^\star(y_i, y_{i-k})} & i - j \in [m] \\ 0 & \text{otherwise} \end{cases} \qquad \mathcal{A}^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ * & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ * & * & 0 & 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \gamma_{T-1}(m) & \dots & \gamma_{T-1}(2) & \gamma_{T-1}(1) & 0 & \dots \\ 0 & \dots & 0 & \gamma_T(m) & \dots & \gamma_T(2) & \gamma_T(1) & 0 \end{pmatrix}$$

where $\hat{m} = \min(i - 1, m)$. For positions $i > m$, where the MTD model is defined, the attention mechanism thus computes $\mathcal{A}^{(1)}_{ij} = \gamma_i(i - j)$ for the active lags ($1 \le i - j \le m$). This expression is precisely the posterior responsibility $\gamma_i(g)$ from Equation 6, where the lag is $g = i - j$. For earlier steps ($1 < i \le m$), the attention mechanism computes the natural counterpart by normalizing over the $i - 1$ available lags. The resulting attention matrix $\mathcal{A}^{(1)}$ has a specific banded lower-triangular structure. The output of this layer, $\hat{\boldsymbol{h}}^{(1)}_i$, is the attention-weighted sum over the concatenated value vectors. Given that the attention weights $\mathcal{A}_{ij}$ compute the posterior responsibilities $\gamma_i(g)$ (for $i > m$), and $\boldsymbol{R}^{(1)}_V$ embeds the lag $g = i - j$ as a one-hot vector $\boldsymbol{e}_g$, the output for a position $i > m$ is:

$$\hat{\boldsymbol{h}}^{(1)}_i = \sum_{j=1}^{i-1} \mathcal{A}_{ij} \operatorname{Concat}(\boldsymbol{e}_{y_j}, \boldsymbol{r}'^{(1)}_{ij}) = \sum_{g=1}^{m} \gamma_i(g) \operatorname{Concat}(\boldsymbol{e}_{y_{i-g}}, \boldsymbol{e}_g)\,. \tag{8}$$

This output is a convex combination, weighted by the posterior responsibilities, of vectors that each concatenate two pieces of information: the one-hot encoding of the token at a given lag ($\boldsymbol{e}_{y_{i-g}}$) and the one-hot encoding of $\boldsymbol{e}_g$ of the lag itself ($g$):

$$\hat{\boldsymbol{h}}^{(1)}_i = \underbrace{\gamma_i(m) \begin{pmatrix} \boldsymbol{e}_{y_{i-m}} \\ 0 \\ \vdots \\ 1 \end{pmatrix}}_{\text{Lag } m} + \cdots + \underbrace{\gamma_i(2) \begin{pmatrix} \boldsymbol{e}_{y_{i-2}} \\ 0 \\ 1 \\ \vdots \end{pmatrix}}_{\text{Lag 2}} + \underbrace{\gamma_i(1) \begin{pmatrix} \boldsymbol{e}_{y_{i-1}} \\ 1 \\ 0 \\ \vdots \end{pmatrix}}_{\text{Lag 1}} = \begin{pmatrix} \sum_{g=1}^{m} \gamma_i(g) \boldsymbol{e}_{y_{i-g}} \\ \gamma_i(1) \\ \gamma_i(2) \\ \vdots \\ \gamma_i(m) \end{pmatrix}$$

In essence, the top part of $\hat{\boldsymbol{h}}^{(1)}_i$ contains a weighted sum of past tokens (which is not be used), while its bottom part explicitly stores the vector of posterior responsibilities with the value $\gamma_i(g)$ for lag $g$ stored at the $g$-th position within this second block (i.e., $\gamma_i(1)$ is first, followed by $\gamma_i(2)$, etc.).

**Layer 2, Summing Responsibilities:** The second layer sums along the sequence the responsibility vectors computed in Layer 1, for tokens at positions $i > m$. This is achieved by setting the content-based attention to zero ($\boldsymbol{W}^{(2)}_A = \boldsymbol{0}$) and the value-rpe matrix to zero ($\boldsymbol{R}^{(2)}_V = \boldsymbol{0}$). The mechanism relies on the content-position interaction $(\boldsymbol{h}^{(1)}_i)^\top \boldsymbol{r}^{(2)}_{ij}$. Crucially, the input vector $\boldsymbol{h}^{(1)}_i = \operatorname{Concat}(\boldsymbol{e}_{y_i}, \hat{\boldsymbol{h}}^{(1)}_i)$ retains in the residual stream the one-hot embedding $\boldsymbol{e}_{y_i}$ of the current token in its first $q$ dimensions. The RPE table $\boldsymbol{R}^{(2)}_A$ is structured to interact only with this part of the vector, turning the dot product into a fixed bias:

$$\boldsymbol{W}^{(2)}_A = \boldsymbol{0}_{d_1 \times d_1}, \quad (\boldsymbol{R}^{(2)}_A)_{k,:} = \begin{cases} \boldsymbol{0}_{1 \times d_1} & \text{if } 1 \le k \le T - m \\ [-\delta_2 \cdot \boldsymbol{1}_{1\times q}, \boldsymbol{0}_{1\times(q+m)}] & \text{otherwise} \end{cases}, \quad \boldsymbol{R}^{(2)}_V = \boldsymbol{0}$$

The attention score for the final query at $i = T$ therefore simplifies to:

$$e_{Tj} = (\boldsymbol{h}^{(1)}_T)^\top \boldsymbol{r}^{(2)}_{Tj} = \begin{cases} (\boldsymbol{h}^{(1)}_T)^\top \begin{pmatrix} -\delta_2 \cdot \boldsymbol{1}_q \\ \boldsymbol{0}_{q+m} \end{pmatrix} = -\delta_2 \cdot (\boldsymbol{e}^\top_{y_T} \boldsymbol{1}_q) = -\delta_2 & \text{if } 1 \le j \le m \\ (\boldsymbol{h}^{(1)}_T)^\top \boldsymbol{0}_{d_1} = 0 & \text{otherwise} \end{cases}$$

In the limit $\delta_2 \to \infty$, the softmax places uniform attention only on the keys where the score is not $-\infty$. This results in uniform attention weights $\mathcal{A}_{Tj} = 1/(T - m)$ for $j \in [m + 1, T]$, and zero

otherwise. The layer's output for the final token is therefore the exact average of the desired vectors:

$$\hat{\boldsymbol{h}}_T^{(2)} = \sum_{j=1}^{T} \mathcal{A}_{Tj} \operatorname{Concat}(\boldsymbol{h}_j^{(1)}, \boldsymbol{r}'_{Tj}) = \frac{1}{T-m} \sum_{j=m+1}^{T} \operatorname{Concat}(\boldsymbol{h}_j^{(1)}, \mathbf{0}) = \frac{1}{T-m} \begin{pmatrix} \sum_{j=m+1}^{T} \left( \sum_{g=1}^{m} \gamma_j(g) \boldsymbol{e}_{y_{j-g}} \right) \\ \sum_{j=m+1}^{T} \gamma_j(1) \\ \sum_{j=m+1}^{T} \gamma_j(2) \\ \vdots \\ \sum_{j=m+1}^{T} \gamma_j(m) \\ \hline \mathbf{0} \end{pmatrix}$$

Defining $\boldsymbol{\Gamma}_j = (\gamma_j(1), \gamma_j(2), \ldots, \gamma_j(m))^\top$; the $m$-dimensional sub-block $\hat{\boldsymbol{h}}_{q+1:q+m}^{(2)} = \sum_j \boldsymbol{\Gamma}_j$ contains the average responsibility vectors, $\frac{1}{T-m} \sum_{j=m+1}^{T} \boldsymbol{\gamma}_j$, which is exactly the quantity needed to implement the one-step MD estimator in Prop. 2.

**Layer 3, Final Predictive Weights:** The third and final layer uses the averaged responsibilities computed in Layer 2 to produce the final predictive weights, $\tilde{\boldsymbol{\lambda}}$, which correspond to the one-step MD estimate from Prop. 2. This is accomplished by using the RPE table, $\boldsymbol{R}_A^{(3)}$, to perform a selective dot product. The query vector for the final token, $\boldsymbol{h}_T^{(2)}$, contains the vector of averaged responsibilities in its final $m$ coordinates. The RPE vectors in $\boldsymbol{R}_A^{(3)}$ are constructed as scaled one-hot vectors that align with this sub-block, effectively using the dot product to "read out" the corresponding averaged responsibility. Similarly to layer 2, to only have non-zero attention only at the positions corresponding to the $m$ lags, we use the one-hot embedding of the current token $e_{y_i}$ from the residual stream $\boldsymbol{h}_i^{(2)} = \operatorname{Concat}(\boldsymbol{e}_{y_i}, \hat{\boldsymbol{h}}_i^{(1)}, \hat{\boldsymbol{h}}_i^{(2)})$ to add a fixed bias $\delta_3$ which drives the softmax to zero in the limit. The content-based attention and value-rpe are again disabled:

$$\begin{aligned} \boldsymbol{W}_A^{(3)} &= \mathbf{0}_{d_2 \times d_2} \\ \boldsymbol{R}_V^{(3)} &= \mathbf{0} \end{aligned}, \qquad (\boldsymbol{R}_A^{(3)})_{k,:} = \begin{cases} [\overbrace{+\delta_3 \cdot \mathbf{1}_q^\top}^{\text{on } \boldsymbol{h}^{(0)}}, \overbrace{\mathbf{0}_{q+m}^\top}^{\text{on } \hat{\boldsymbol{h}}^{(1)}}, \overbrace{\mathbf{0}_q^\top}^{\text{on } \hat{\boldsymbol{h}}_{1:q}^{(2)}}, \overbrace{\beta \cdot \boldsymbol{e}_k^\top}^{\text{on } \boldsymbol{\Gamma}}, \overbrace{\mathbf{0}_q^\top}^{\text{on } \hat{\boldsymbol{h}}_{q+m+1:d_2}^{(2)}}] & \text{if } k \in [m] \\ [-\delta_3 \cdot \mathbf{1}_q^\top, \mathbf{0}_{q+m}^\top, \mathbf{0}_q^\top, \mathbf{0}_m^\top, \mathbf{0}_q^\top] & \text{otherwise} \end{cases}$$

Here, $\boldsymbol{e}_k$ is a one-hot vector in $\mathbb{R}^{d_2}$ that selects the coordinate corresponding to the $k$-th responsibility in $\boldsymbol{h}_T^{(2)}$, and $\beta$ is the learnable scaled learning rate. Visually, the RPE table $\boldsymbol{R}_A^{(3)}$ is a sparse matrix of scaled one-hot vectors:

$$\boldsymbol{R}_A^{(3)\top} = \begin{array}{c} \\ \boldsymbol{h}^{(0)} \\ \hat{\boldsymbol{h}}^{(1)} \\ \hat{\boldsymbol{h}}_{1:q}^{(2)} \\ \boldsymbol{\Gamma} \\ \hat{\boldsymbol{h}}_{q+m+1:d_2}^{(2)} \end{array} \begin{array}{c} \begin{array}{cccc} [1, \cdots, m] & \cdots & \cdots & T \end{array} \\ \left( \begin{array}{cccc} +\delta_3 \cdot \mathbf{1}_q, \cdots, +\delta_3 \cdot \mathbf{1}_q & -\delta_3 \cdot \mathbf{1}_q & -\delta_3 \cdot \mathbf{1}_q & -\delta_3 \cdot \mathbf{1}_q \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ [\beta \boldsymbol{e}_1 \ \beta \boldsymbol{e}_2 \cdots \beta \boldsymbol{e}_m] & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array} \right) \end{array}$$

Since $W_A^{(3)} = \mathbf{0}$, the score simplifies to the content-position interaction $(\boldsymbol{h}_T^{(2)})^\top \boldsymbol{r}_{Tj}^{(3)}$ and $\boldsymbol{r}_{Tj}^{(3)}$ is constructed to be non-zero only for relative positions $g = T - j + 1 \in [1, m]$. For these lags, the RPE is a scaled one-hot vector that acts as a selector, using the dot product to extract the corresponding averaged responsibility stored in $\boldsymbol{h}_T^{(2)}$:

$$e_{Tj} = (\boldsymbol{h}_T^{(2)})^\top \boldsymbol{r}_{Tj}^{(3)} = \begin{cases} (\boldsymbol{h}_T^{(0)})^\top (\delta_3 \mathbf{1}_q) + (\boldsymbol{\Gamma})_T^\top (\beta \boldsymbol{e}_g) \\ -\delta_3 \cdot (\boldsymbol{e}_{y_T}^\top \mathbf{1}_q) \end{cases} = \begin{cases} +\delta_3 + \beta \cdot \frac{\sum_{i=m+1}^{T} \gamma_i(g)}{T-m} & \text{if } g \in [m] \\ -\delta_3 & \text{otherwise.} \end{cases}$$

The attention scores for the final token, $e_{T,:}$, becomes the scaled averaged responsibility:

$$\mathbf{e}_{T,:} = \Big( -\delta_3, \ldots, -\delta_3, \underbrace{+\delta_3 + \beta \frac{\sum_{i=m+1}^{T} \gamma_i(m)}{T-m}}_{\text{pos } T-m+1}, \ldots, \underbrace{+\delta_3 + \beta \frac{\sum_{i=m+1}^{T} \gamma_i(2)}{T-m}}_{\text{pos } T-1}, \underbrace{+\delta_3 + \beta \frac{\sum_{i=m+1}^{T} \gamma_i(1)}{T-m}}_{\text{pos } T} \Big)$$

After applying the softmax and in the limit of large $\delta_3$, the attention weights for the last token compute the MD estimate of the mixture weights $\tilde{\lambda}_g$ and place them at the correct positions:

$$\lim_{\delta_3 \to \infty} \mathcal{A}_{T,T-g} = \frac{\exp\left( \beta \frac{\sum_{i=m+1}^{T} \gamma_i(g)}{T-m} \right)}{\sum_{k=1}^{m} \exp\left( \beta \frac{\sum_{i=m+1}^{T} \gamma_i(k)}{T-m} \right)} \qquad \lim_{\delta_3 \to \infty} \mathcal{A}_{T,:} = \Big( \ldots, 0, \underbrace{\tilde{\lambda}_m}_{T-m+1}, \ldots, \underbrace{\tilde{\lambda}_2}_{T-1}, \underbrace{\tilde{\lambda}_1}_{T} \Big).$$

This final attention operation is the core of the estimation process. It computes the mixture weights $\tilde{\boldsymbol{\lambda}}$, which serve as the in-context estimates of token importance, thus realizing the mechanism of in-context learning this work seeks to understand.

**Output Layer:** The final step of the construction is to apply the output matrix $\widetilde{\boldsymbol{W}}_O$ to the final hidden state $\boldsymbol{h}_T^{(3)}$ to produce the predictive distribution over the next token. The matrix $\widetilde{\boldsymbol{W}}_O$ learns the known transition matrix $\boldsymbol{\pi}^\star$ and selectively applies it to the embedding of the last token. The output of the third attention layer, $\hat{\boldsymbol{h}}_T^{(3)}$, is a weighted sum of the hidden states from the second layer, where the weights are the estimated mixture weights $\tilde{\lambda}_g$: $\hat{\boldsymbol{h}}_T^{(3)} = \sum_{j=1}^{T} \mathcal{A}_{Tj}^{(3)} \boldsymbol{h}_j^{(2)} = \sum_{g=1}^{m} \tilde{\lambda}_g \boldsymbol{h}_{T-g}^{(2)}$. The first $q$ components of any hidden state $\boldsymbol{h}_j^{(k)}$, due to the residual stream, simply contain the original input $\boldsymbol{h}_j^{(0)} = \boldsymbol{e}_{y_j}$. Consequently, the first $q$ components of $\hat{\boldsymbol{h}}_T^{(3)}$ are a $\tilde{\lambda}$-weighted combination of the one-hot embeddings of the relevant past tokens: $(\hat{\boldsymbol{h}}_T^{(3)})_{1:q} = \sum_{g=1}^{m} \tilde{\lambda}_g (\boldsymbol{h}_{T-g}^{(2)})_{1:q} = \sum_{g=1}^{m} \tilde{\lambda}_g \boldsymbol{e}_{y_{T-g}}$. The full hidden stat is $\boldsymbol{h}_T^{(3)} = \mathrm{Concat}(\boldsymbol{h}_T^{(0)}, \hat{\boldsymbol{h}}_T^{(1)}, \hat{\boldsymbol{h}}_T^{(2)}, \hat{\boldsymbol{h}}_T^{(3)})$ and the output matrix $\widetilde{\boldsymbol{W}}_O \in \mathbb{R}^{q \times d_3}$ is structured to ignore all preceding blocks and operate only on the first $q$ components of the final block, $\hat{\boldsymbol{h}}_T^{(3)}$. This is achieved by storing the transition matrix, $\boldsymbol{\pi}^{\star\top}$, in the corresponding sub-block:

$$\widetilde{\boldsymbol{W}}_O = \big( \overset{\text{from } \boldsymbol{h}_T^{(0)}}{\mathbf{0}_{q\times q}} \;\big|\; \overset{\text{from } \hat{\boldsymbol{h}}_T^{(1)}}{\mathbf{0}_{q\times(q+m)}} \;\big|\; \overset{\text{from } \hat{\boldsymbol{h}}_T^{(2)}}{\mathbf{0}_{q\times(2q+2m)}} \;\big|\; [\boldsymbol{\pi}_{q\times q}^{\star\top} \quad \overset{\text{from } \hat{\boldsymbol{h}}_T^{(3)}}{\mathbf{0}_{q\times(3q+4m)}}] \big) \,.$$

Applying this matrix to the fully expanded final hidden state yields the predictive distribution:

$$\tilde{\mathcal{T}}(\boldsymbol{y}_{1:T})_T = [\boldsymbol{\pi}^{\star\top} \quad \mathbf{0}]\, \hat{\boldsymbol{h}}_T^{(3)} = \boldsymbol{\pi}^{\star\top} (\hat{\boldsymbol{h}}_T^{(3)})_{1:q} = \boldsymbol{\pi}^{\star\top} \left( \sum_{g=1}^{m} \tilde{\lambda}_g \boldsymbol{e}_{y_{T-g}} \right) = \sum_{g=1}^{m} \tilde{\lambda}_g \boldsymbol{\pi}^\star(y_{T-g}, :) \,.$$

This final vector is exactly the predictive distribution from Proposition 3, completing the proof.

## 5 Why One-Step Mirror Descent Works

This section provides a theoretical analysis of the one-step MD estimator to formally justify its success. We prove that a single MD update, initialized from a uniform prior, corresponds to an estimator that is an approximation of the Bayesian posterior mean. Our results therefore, elucidate the mechanism by which this simple, non-iterative procedure achieves good performance.

**One-Step MD as a First-Order Bayesian Approximation:** We establish a theoretical connection between the one-step Mirror Descent (MD) estimator and the Bayesian posterior mean. We show that their first-order Taylor expansions around the state of no evidence coincide up to a scalar constant. This result justifies interpreting the one-step MD estimator as a principled approximation to the Bayes-optimal predictor, especially in low-data regimes. The analysis hinges on treating both estimators as functions of the log-likelihood gradient evaluated at the center of the simplex, $\boldsymbol{g} \coloneqq \nabla_{\boldsymbol{\lambda}} \ell(\boldsymbol{\lambda}^{(0)})$, and expanding them around the point of no evidence, $\boldsymbol{g} = \mathbf{0}$.

**Theorem 1** (First-Order Equivalence of the Estimators)**.** *Let $\hat{\boldsymbol{\lambda}}^{MD}(\boldsymbol{g}; \eta)$ be the one-step MD estimator with learning rate $\eta$, and let $\hat{\boldsymbol{\lambda}}^{Bayes}(\boldsymbol{g})$ be the Bayesian posterior mean under the linearized likelihood. The two estimators are first-order equivalent at $\boldsymbol{g} = \mathbf{0}$ for $\eta = \frac{1}{m+1}$.*

**Learning-rate scaling via a Lipschitz (smoothness) constant:** We established a first-order equivalence between the one-step MD estimator and the Bayesian posterior mean at $\boldsymbol{g} = \mathbf{0}$ (the "no-evidence" regime). For a sequence of length $T$, however, the gradient norm $|\boldsymbol{g}|$ scales with $T$ see App. J, raising the question of how to scale the learning rate of $\hat{\boldsymbol{\lambda}}^{\mathrm{MD}} = \mathrm{softmax}(\eta \boldsymbol{g})$ with $T$. Mirror Descent theory suggests choosing the learning rate inversely proportional to the relative smoothness constant $L_{\mathrm{rel}}$ (Bauschke et al., 2017). Because our MD update uses a negative entropy regularizer, smoothness is defined w.r.t. the KL-divergence. Convergence requires $\eta \leq 1/L_{\mathrm{rel}}$. We now bound this constant and find exactly the scaling implemented in the Transformer in Prop. 3.

**Theorem 2** (Relative Smoothness and $\eta$ scaling at the uniform mixture)**.** *At the uniform vector $\boldsymbol{\lambda} = (1/m, \ldots, 1/m)$, the loss $f(\boldsymbol{\lambda}) = -\ell(\boldsymbol{\lambda})$ is $L_{rel}$-smooth relative to the KL-divergence, with*

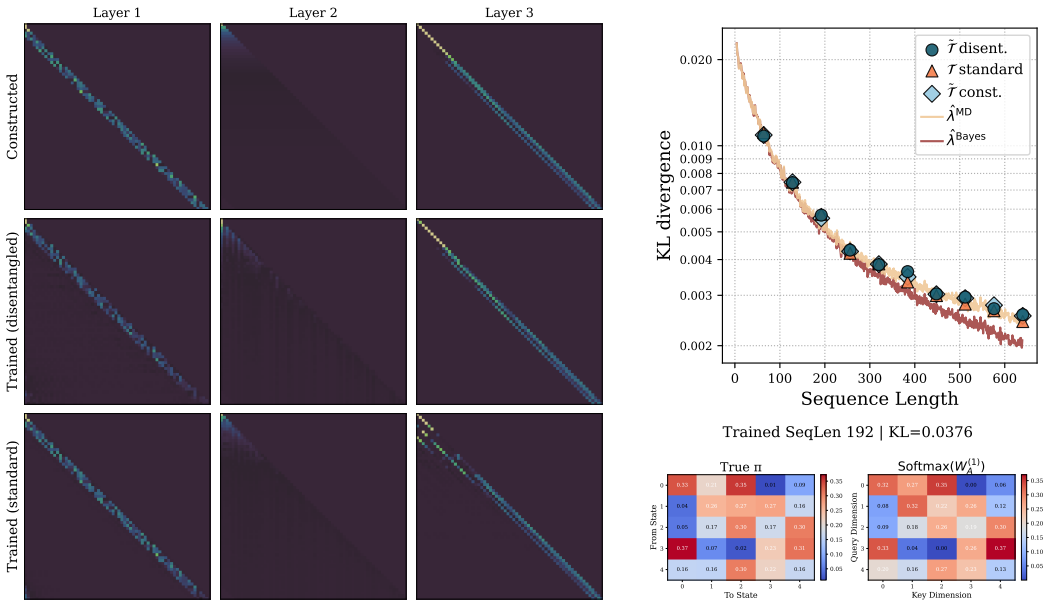


Figure 4: **Comparison of Trained and Constructed Transformers. Left:** Attention maps of the trained transformer (disentanged and standard) versus our theoretical construction (seq. length 64). **Right (top):** KL divergence to the ground truth transition probabilities for the trained transformers, the constructed transformer, and the one-step MD estimator across sequence lengths. **Right (bottom):** First-layer attention softmax Softmax($\boldsymbol{W}_1^A$) vs. true transitions matrix $\pi^\star$ for a trained model.

$L_{rel} \leq (T-m)\,m^2$. *Consequently, the stable MD step-size rule* $\eta \leq \frac{1}{L_{rel}}$ *yields the asymptotic scaling* $\eta = \Theta\left(\frac{1}{T}\right)$ *for fixed* $m$.

**Beyond the Local Approximation, The Implicit Regularization of Mirror Descent:** The first-order equivalence in Theorem 3 holds only for short sequences, where the log-likelihood gradient is small. For longer sequences, neglected higher-order terms become significant, and the one-step estimator diverges from the Bayesian mean. Empirically, however, a few additional Mirror Descent substantially reduces this gap (see Figure 5). In this regime, iterating MD to convergence yields the suboptimal MLE, while early stopping provides a much closer match to the Bayesian mean. We propose that this effect arises from implicit regularization. Specifically, early-stopped MD approximately solves an entropy-regularized optimization problem of the form $\min_\lambda -\ell(\lambda) + \gamma H(\lambda)$ with the iterates tracking the corresponding regularization path (Suggala et al., 2018). Along this path, performance on par with the Bayes-optimal estimator is achieved for an appropriate choice of regularization $\gamma$ (see Figure. 3). Thus, early stopping is effectively equivalent to selecting this favorable point on the entropy-regularized path, avoiding the suboptimal MLE.

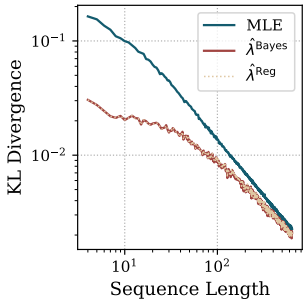


Figure 3: **Regularized Estimator.** Comparison with Bayes and MLE estimators.

## 6 Experiments

We report empirical validation of the main claims and additional experimental details in App.B as well as additional experiments in App.D.

**Setup:** We train 3-layer disentangled transformers $\tilde{\mathcal{T}}_{\text{disent}}$ with single head with learned relative positional and one-hot semantic embeddings as well as 3-layer standard transformers $\mathcal{T}_{\text{standard}}$ (no disentanglement) with single head attention with standard parameterization given by Query-Key-Value with learned relative positional and learned semantic embeddings. We sample a fixed $q$, $m$, $\boldsymbol{\pi}$

and at each iteration we sample a set of $\{\boldsymbol{\lambda_i}\}_{i=1}^{B}$ with $B = 128$ the batch size and $\boldsymbol{\lambda_i} \sim \text{Dir}(\boldsymbol{\alpha})$ and generate $B$ sequences according to the MTD model. We train the model for $10^6$ iterations using the Adam optimizer and MSE loss over the last token in the sequence for various sequence lengths. The learning rate is $10^{-2}$ and decaying it at $0.5$ and $0.75$ of the iterations by a factor of $0.1$.

**Results for one-step MD:** We plot the KL divergence with the ground truth transition probabilities between the trained transformer $\tilde{\mathcal{T}}_{\text{disent.}}$ and $\mathcal{T}_{\text{standard.}}$ compared to our theoretical construction $\tilde{\mathcal{T}}_{\text{constr.}}$, the one-step MD estimator $\hat{\lambda}^{\text{MD}}$ and the optimal-Bayes estimator $\hat{\lambda}^{\text{Bayes}}$ (we compute it via MCMC see App. F.1) for various sequence lengths in Figure 4 (right). We observe that both the disentangled and standard trained transformers match the performance of the theoretical construction and the one-step MD estimator: for small sequence lengths, the latter serves as a good proxy for the optimal Bayes estimator, validating our theoretical result in Theorem 3, whereas for longer sequences it becomes suboptimal. By inspecting the attention maps of the trained transformers (both standard and disentangled) in Figure 4 (left) we can see that they actually learn to extract the responsibilities $\gamma(g)_i$ as expected from our construction (for all three models, the locations of low and high attention entries, and the diagonal structure induced by the MTD order, are closely aligned). To further validate if the attention matrix in the first layer actually learns the ground truth transition matrix $\pi^\star$ we plot the heatmap of the first attention softmax $\text{Softmax}(\boldsymbol{W}_1^A)$ vs. $\pi^\star$ as well compute the averge row-wise KL divergence between the two matrices Figure 4 (bottom-right) more results in App. D. For both the $\hat{\lambda}^{\text{MD}}$ and $\tilde{\mathcal{T}}_{\text{constr.}}$ we tune the learning rate $\beta$ and $\eta$ via grid search to minimize the KL divergence (see Section B.1 for more details).

**Results multi-step MD:** To further validate the hypothesis that Transformers can learn to implement Mirror Descent to solve the task, we plot the KL divergence for the 5-layer trained transformer $\tilde{\mathcal{T}}$train. compared to the multi-step MD estimator $\hat{\lambda}^{\text{MD},k=i}$ where $i$ is the number of steps for various sequence lengths in Figure 5. We observe that the trained transformers match the performance of 2-step of Mirror Descent. Importantly, for longer sequences where the difference between the 2-step MD and the Bayes-optimal solution becomes significant, the transformer still matches the 2-step performance thus confirming that it is implementing an estimator which matches at least in performance the 2-step MD. While this evidence offers useful insights into how the transformer behaves, it does not constitute direct evidence that it is implementing multi-step MD. Extending our construction to the multi-step MD setting is nontrivial and we leave this to future work Empirically, we find that a 5-layer transformer suffices to match the performance of two steps, suggesting that the responsibility computed from the first step might be reused to approximate the second.

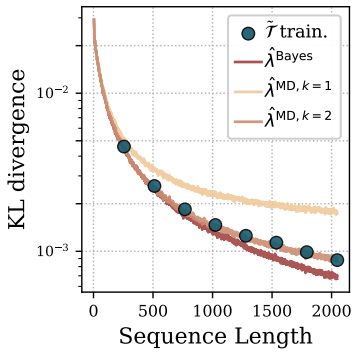


Figure 5: **Multi-Step MD vs. 5-Layer Transformer.** Comparison of the KL divergence to the Bayesian posterior mean.

## 7 Conclusions

This work identifies Mirror Descent as the core algorithm that transformers implement for in-context learning of latent mixture weights. We introduced a novel framework using Mixture of Transition Distribution (MTD) models to frame the inference of token importance as an in-context, latent variable estimation task. Within this framework, we provided a constructive proof that a 3-layer transformer can exactly implement one-step of Mirror Descent, and we showed empirically that deeper models learn to approximate multiple steps of the algorithm. Our theoretical analysis provides insight into te one-step MD estimator, establishing its connection to the Bayes-optimal estimator and deriving practical scaling laws for stable learning. Taken together, our findings extend the gradient-based interpretation of ICL to sequential tasks over discrete domains, providing a new algorithmic explanation for how transformers reason over latent structures.

## REFERENCES


Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2022.

Adrian Raftery André Berchtold. The Mixture Transition Distribution Model for High-Order Markov Chains and Non-Gaussian Time Series. *Statistical Science*, 17(3):328 – 356, 2002. doi: 10.1214/ss/1042727943.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=liMSqUuVg9.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36, 2023b.

Heinz H Bauschke, Jérôme Bolte, and Marc Teboulle. A descent lemma beyond lipschitz gradient continuity: first-order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348, 2017.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 2023a.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. Birth of a transformer: A memory viewpoint. *Advances in Neural Information Processing Systems*, 36, 2023b.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Stephanie Chan, Adam Santoro, Andrew Lampinen, Jane Wang, Aaditya Singh, Pierre Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. *Advances in neural information processing systems*, 35:18878–18891, 2022.

Siyu Chen, Heejune Sheen, Tianhao Wang, and Zhuoran Yang. Unveiling induction heads: Provable training dynamics and feature learning in transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=4fN2REs0Ma.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=89ia77nZ8u.

Francesco D'Angelo, Francesco Croce, and Nicolas Flammarion. Selective induction heads: How transformers select causal structures in context. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bnJgzAQjWf.

Ezra Edelman, Nikolaos Tsilivis, Benjamin Edelman, Eran Malach, and Surbhi Goel. The evolution of statistical induction heads: In-context learning markov chains. *Advances in Neural Information Processing Systems*, 37:64273–64311, 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

Dan Friedman, Alexander Wettig, and Danqi Chen. Learning transformer programs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Pe9WxkN8Ff.

Deqing Fu, Tian qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn to achieve second-order convergence rates for in-context linear regression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L8h6cozcbn.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Jiachen Hu, Qinghua Liu, and Chi Jin. On limitation of transformer for learning hmms. *arXiv preprint arXiv:2406.04089*, 2024.

M Emrullah Ildiz, Yixiao Huang, Yingcong Li, Ankit Singh Rawat, and Samet Oymak. From self-attention to markov models: Unveiling the dynamics of generative transformers. *arXiv preprint arXiv:2402.13512*, 2024.

Yuchen Li, Yuanzhi Li, and Andrej Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19689–19729. PMLR, 23–29 Jul 2023.

Arvind Mahankali, Tatsunori B Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. In *The Twelfth International Conference on Learning Representations*, 2024.

Ashok Vardhan Makkuva, Marco Bondaschi, Adway Girish, Alliot Nagle, Martin Jaggi, Hyeji Kim, and Michael Gastpar. Attention with markov: A framework for principled analysis of transformers via markov chains. *arXiv preprint arXiv:2402.04161*, 2024.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

Timothy Nguyen. Understanding transformers via n-gram statistics. *arXiv preprint arXiv:2407.12034*, 2024.

Eshaan Nichani, Alex Damian, and Jason D. Lee. How transformers learn causal structure with gradient descent. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=jNM4imlHZv.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

Reese Pathak, Rajat Sen, Weihao Kong, and Abhimanyu Das. Transformers can optimally learn regression mixture models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sLkj91HIZU.

Adrian E Raftery. A model for high-order markov chains. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 47(3):528–539, 1985.

Nived Rajaraman, Marco Bondaschi, Kannan Ramchandran, Michael Gastpar, and Ashok Vardhan Makkuva. Transformers on markov data: Constant depth suffices. *arXiv preprint arXiv:2407.17686*, 2024.

Allan Raventos, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Michael Eli Sander, Raja Giryes, Taiji Suzuki, Mathieu Blondel, and Gabriel Peyré. How do transformers perform in-context autoregressive learning ? In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=kZbTkpnafR.

Adam Shai, Paul M. Riechers, Lucas Teixeira, Alexander Gietelink Oldenziel, and Sarah Marzen. Transformers represent belief state geometry in their residual stream. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=YIB7REL8UC.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *North American Chapter of the Association for Computational Linguistics*, 2018. URL https://api.semanticscholar.org/CorpusID:3725815.

Aaditya K Singh, Ted Moskovitz, Felix Hill, Stephanie C.Y. Chan, and Andrew M Saxe. What needs to go right for an induction head? a mechanistic study of in-context learning circuits and their formation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=O8rrXl71D5.

Arun Suggala, Adarsh Prasad, and Pradeep K Ravikumar. Connecting optimization and regularization paths. *Advances in Neural Information Processing Systems*, 31, 2018.

Anej Svete and Ryan Cotterell. Transformers can represent $n$-gram language models. *arXiv preprint arXiv:2404.14994*, 2024.

Aditya Varre, Gizem Yüce, and Nicolas Flammarion. Learning in-context $n$-grams with transformers: Sub-$n$-grams are near-stationary points. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=OMwdvGDeHL.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023a.

Johannes Von Oswald, Eyvind Niklasson, Maximilian Schlegel, Seijin Kobayashi, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Max Vladymyrov, Razvan Pascanu, et al. Uncovering mesa-optimization algorithms in transformers. *arXiv preprint arXiv:2309.05858*, 2023b.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022.

Steve Yadlowsky, Lyric Doshi, and Nilesh Tripuraneni. Pretraining data mixtures enable narrow model selection capabilities in transformer models. *arXiv preprint arXiv:2311.00871*, 2023.

Oğuz Kaan Yüksel and Nicolas Flammarion. On the sample complexity of next-token prediction. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025. URL https://openreview.net/forum?id=eJkNMwzZzy.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023.

Nicolas Zucchet, Francesco d'Angelo, Andrew K Lampinen, and Stephanie CY Chan. The emergence of sparse attention: impact of data distribution and benefits of repetition. *arXiv preprint arXiv:2505.17863*, 2025.

# A NOTATION

Throughout this paper, we use non-bold letters for scalars (e.g., $\eta, \alpha$), lowercase bold letters for vectors (e.g., $\boldsymbol{h}, \boldsymbol{\lambda}$), and uppercase bold letters for matrices (e.g., $\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\pi}$). The $i$-th element of a vector $\boldsymbol{v}$ is denoted by $v_i$, and the vector at position $i$ in a sequence of vectors $\boldsymbol{H}$ is written as $\boldsymbol{h}_i$. The element in the $i$-th row and $j$-th column of a matrix $\boldsymbol{A}$ is $A_{ij}$, and its $i$-th row vector is $\boldsymbol{A}_{i,:}$. We use $\mathbf{1}$ and $\mathbf{0}$ to denote vectors or matrices of ones and zeros, respectively, with dimensions inferred from context. We use $\boldsymbol{e}_k$ to denote a one-hot vector with a one at the $k$-th position; its dimensionality is specified or clear from context. The set of integers $\{1, \ldots, m\}$ is denoted by $[m]$. The probability simplex in $\mathbb{R}^m$ is denoted by $\Delta_{m-1}$. The operator $\mathrm{Concat}(\cdot, \cdot)$ denotes the vertical concatenation of vectors or matrices. For vectors $\boldsymbol{a} \in \mathbb{R}^{d_a}$ and $\boldsymbol{b} \in \mathbb{R}^{d_b}$, their concatenation results in a vector in $\mathbb{R}^{d_a+d_b}$. For matrices $\boldsymbol{A} \in \mathbb{R}^{d_A \times T}$ and $\boldsymbol{B} \in \mathbb{R}^{d_B \times T}$ with the same number of columns, $\mathrm{Concat}(\boldsymbol{A}, \boldsymbol{B})$ is the block matrix $\left(\begin{smallmatrix} \boldsymbol{A} \\ \boldsymbol{B} \end{smallmatrix}\right) \in \mathbb{R}^{(d_A+d_B)\times T}$. Superscripts in parentheses, such as $\boldsymbol{H}^{(l)}$, are used to index the layers of the Transformer. In the context of Transformer relative positional encodings, we use 1-based indexing for token positions $i, j \in [T]$. The relative position is mapped to a lookup table index $k = i - j + 1$. The approximations will be expressed using Landau notation, where a vector function $\boldsymbol{f}(\boldsymbol{g}) = O(\|\boldsymbol{g}\|^p)$ signifies that $\|\boldsymbol{f}(\boldsymbol{g})\| \leq C\|\boldsymbol{g}\|^p$ for some constant $C$ in a neighborhood of $\boldsymbol{g} = \mathbf{0}$.

## A.1 DIMENSIONALITY OF THE TRANSFORMER CONSTRUCTION

The disentangled Transformer architecture results in a hidden state that grows with each layer. The following table provides a summary of the dimensions at each stage of the construction. Note that the input to layer $l$ is $\boldsymbol{h}^{(l-1)}$, and its output is $\boldsymbol{h}^{(l)}$, which is formed by concatenating the input with the result of the attention mechanism, $\hat{\boldsymbol{h}}^{(l)}$. We use $q$ for the alphabet size and $m$ for the MTD model order. For this construction, the RPE value dimension $d_R$ is set to $m$.

Table 1: Dimensionality of Hidden States in the Disentangled Transformer

| **Layer** | **Description** | **Attention Output $\hat{\boldsymbol{h}}^{(l)}$** | **Concatenated Hidden State $\boldsymbol{h}^{(l)}$** |
|---|---|---|---|
| 0 (Input) | Token Embeddings | — | $\boldsymbol{h}^{(0)} \in \mathbb{R}^q$ |
| 1 | Responsibilities | $\hat{\boldsymbol{h}}^{(1)} \in \mathbb{R}^{q+m}$ | $\boldsymbol{h}^{(1)} = \mathrm{Concat}(\boldsymbol{h}^{(0)}, \hat{\boldsymbol{h}}^{(1)}) \in \mathbb{R}^{2q+m}$ |
| 2 | Summation | $\hat{\boldsymbol{h}}^{(2)} \in \mathbb{R}^{2q+m}$ | $\boldsymbol{h}^{(2)} = \mathrm{Concat}(\boldsymbol{h}^{(1)}, \hat{\boldsymbol{h}}^{(2)}) \in \mathbb{R}^{4q+2m}$ |
| 3 | Weighting | $\hat{\boldsymbol{h}}^{(3)} \in \mathbb{R}^{4q+2m}$ | $\boldsymbol{h}^{(3)} = \mathrm{Concat}(\boldsymbol{h}^{(2)}, \hat{\boldsymbol{h}}^{(3)}) \in \mathbb{R}^{8q+4m}$ |
| **Final Output Matrix** | | $\widetilde{\boldsymbol{W}}_O \in \mathbb{R}^{q \times (8q+4m)}$ | |

# B EXPERIMENTAL DETAILS

This section provides additional details on the experimental setup, including the hyperparameter settings for all models and estimators used in our empirical validation.

## B.1 HYPERPARAMETER TUNING

For the one-step Mirror Descent estimator ($\hat{\lambda}^{\mathrm{MD}}$) and our theoretical Transformer construction ($\tilde{\mathcal{T}}_{\text{constr.}}$), the learning rate parameters $\eta$ and $\beta$ were not fixed but were tuned to optimize performance. For each sequence length evaluated, we performed a grid search over a range of potential values for $\eta$ and $\beta$. The value that minimized the KL divergence to the true Bayesian posterior mean was selected for the final comparison plots. This ensures that both methods were evaluated under their optimal conditions.

## B.2 PARAMETER SUMMARY

The following table summarizes the key parameters used in our experiments.

Table 2: Summary of Experimental Parameters

| Component | Parameter | Value |
|---|---|---|
| **Data Generation** | | |
| | MTD model order ($m$) | 3,4,5 |
| | Vocabulary size ($q$) | 5 |
| | Sequence Length ($T$) | Varied (64 to 1984) |
| | Dirichlet Prior ($\boldsymbol{\alpha}$) | Uniform ($\alpha_g = 1$ for all $g$) |
| | Transition Matrix ($\boldsymbol{\pi}$) | Rows from Dirichlet ($\boldsymbol{\alpha} = 1$) |
| **Mirror Descent (MD) Estimator** | | |
| | Learning Rate ($\eta$) | Log grid: $[10^{-5}, 10^{-1}]$, 1000 points |
| **Constructed Transformer ($\tilde{\mathcal{T}}_{\text{constr.}}$)** | | |
| | Large Constant ($\delta_1, \delta_2, \delta_3$) | [100,100,100] |
| | Scaled Learning Rate ($\beta$) | Log grid: $[10^{-5}, 10^{-1}]$, 1000 points |
| **Trained Transformer ($\tilde{\mathcal{T}}_{\text{disent.}}$)** | | |
| | Architecture | 3-layer & 5-layer Disentangled Transformer |
| | Attention Heads | 1 |
| | Concatenation | True |
| | Semantic Embeddings | one-hot |
| | Relative Positional Encodings | Learned |
| | RPE value dimension | $m$ |
| | Embedding Dimension | $q$ |
| | QK parametrization | False |
| | Value matrix | False |
| | Head output projection matrix | False |
| | Optimizer | Adam |
| | Learning Rate | $1 \times 10^{-3}$ |
| | LR Schedule | Decay by 0.1 at 50% and 75% of training |
| | Batch Size | 128 |
| | Training Iterations | $1 \times 10^{6}$ |
| | Loss Function | MSE on the last token prediction |
| **Trained Transformer ($\mathcal{T}_{\text{standard.}}$)** | | |
| | Architecture | 3-layer Standard Transformer |
| | Attention Heads | 1 |
| | Concatenation | False |
| | Semantic Embeddings | Learned |
| | Relative Positional Encodings | Learned |
| | Embedding Dimension | 32 |
| | QK parametrization | True |
| | Value matrix | True |
| | Head output projection matrix | True |
| | Optimizer | Adam |
| | Learning Rate | $1 \times 10^{-3}$ |
| | LR Schedule | Decay by 0.1 at 50% and 75% of training |
| | Batch Size | 128 |
| | Training Iterations | $5 \times 10^{5}$ |
| | Loss Function | MSE on the last token prediction |
| **MCMC for Bayes Estimator** | | |
| | Sampler | Gibbs Sampling |
| | Burn-in Iterations | [200] |
| | Number of Samples (K) | [2000] |

## C Related works

**Induction heads and interpretability** The emergence of ICL, as well as the more general ability of transformers to implement algorithms, has been linked to the formation of interpretable computational circuits Elhage et al. (2021) such as induction heads (Olsson et al., 2022), which are woven into sparse attention patterns (Zucchet et al., 2025). The development of these circuits is not monolithic; rather, they emerge in phases through the interaction of simpler subcircuits. Singh et al. (2024), for instance, use a causal framework to identify the key subcircuits whose interplay leads to the sudden formation of induction heads during training. This emergence itself also critically depends on the training data; specific distributional properties, such as burstiness and class imbalance, have been shown to be key drivers of this capability Chan et al. (2022); Zucchet et al. (2025). While much of this understanding comes from reverse-engineering circuits in pretrained models Conmy et al. (2023), a parallel line of research aims to create transformers that are interpretable by design, for example by training models that can be directly decompiled into human-readable programs Friedman et al. (2023).

**In-Context learning and gradient descent** Following initial empirical observations of ICL in transformers (Brown et al., 2020), a significant line of research has sought to understand its underlying mechanisms. Early work demonstrated that transformers can learn simple function classes like linear models in-context (Garg et al., 2022). This led to the hypothesis that transformer layers effectively implement optimization algorithms, with several studies showing they can perform computations analogous to gradient descent for in-context linear regression (Akyürek et al., 2022; Bai et al., 2023b; Von Oswald et al., 2023a;b). This gradient-based view has been extended to higher-order algorithms (Ahn et al., 2023; Fu et al., 2024) and given theoretical grounding, with proofs that gradient flow converges to a transformer that has learned the in-context task (Zhang et al., 2023). From a statistical learning perspective, this process has been formalized as "algorithm learning", where generalization is guaranteed by the algorithmic stability of the learned procedure (Li et al., 2023). Crucially, the emergence of this behavior is not guaranteed; it depends on sufficient pretraining task diversity (Raventos et al., 2023). This algorithmic paradigm also extends to linear autoregressive processes, where transformers have been shown to implement a gradient descent step to learn the transition matrix in-context (Sander et al., 2024).

**In-Context learning, Markov chains and n-gram models** Our work is closely related to the literature analyzing ICL for sequential probabilistic models like n-grams and Markov chains. Foundational work by Yüksel & Flammarion (2025) established formal generalization bounds for next-token prediction on Markovian data, analyzing its sample complexity. Regarding transformers, several mechanistic studies have investigated how they implement learning algorithms for these models. For bigrams, transformers have been shown to develop induction heads that function like associative memories (Bietti et al., 2023b) and accurately compute posterior probabilities from statistical cues (Edelman et al., 2024). For first order Markov chains, Nichani et al. (2024) demonstrated that transformers learn the causal structure with gradient descent and implement induction heads to estimate the transition probabilities in-context, effectively implementing a Bayes-optimal estimator. This analysis was later extended to higher-order chains (Chen et al., 2024), while the work of D'Angelo et al. (2025) shows that transformers can even learn to select the correct Markov causal structure at inference time. Further theoretical results have explored the transformer loss landscape in this setting (Makkuva et al., 2024), characterized in-context n-grams as near-stationary points (Varre et al., 2025), and shown that constant-depth transformers are sufficient to learn k-th order Markov chains (Rajaraman et al., 2024).

**Transformers and sequential models** Beyond specific learning algorithms, a broader line of work has explored the fundamental capabilities and limitations of transformers as sequential models. In terms of representational power, transformers with sparse attention have been shown to be capable of exactly representing any n-gram model (Svete & Cotterell, 2024). However, this expressive power has limits; for instance, transformers may be less effective at learning certain Hidden Markov Models (HMMs) compared to RNNs (Hu et al., 2024). Interestingly, when investigating the internal representations that enable inference in HMMs, Shai et al. (2024) showed that transformers maintain interpretable belief states that are linearly encoded in the residual stream. A strength of transformers is their ability to perform in-context model selection. It has been demonstrated that a single transformer can adaptively choose between different base algorithms or even qualitatively different tasks (e.g., regression vs. classification) based on the prompt (Bai et al., 2023a), effectively

selecting between different function classes in-context (Yadlowsky et al., 2023). While high-level n-gram statistics can approximate transformer predictions, the mechanism for how the correct "rule" is selected in-context remains an open question (Nguyen, 2024).

## D Additional Experiments

In this section we repeat the main-text experiments for different MTD orders, comparing the trained and constructed transformers: in addition to the $m = 4$ case in the main text, we report results for $m = 3$ and $m = 5$. In Figure 6, we plot the KL divergence to the ground-truth transition probabilities across sequence lengths for the trained transformers, the constructed transformer, and the one-step MD estimator (orders 3 and 5). In Figure 7, we instead compare the learned first-layer attention (softmax) to the true transition matrices for orders 3 and 5, with the average row-wise KL divergence reported directly in each panel. Finally, Figure 8 reports attention grids for the trained and constructed transformers (disentangled) at sequence length 64 for MTD orders $m = 3$ and $m = 5$, analogous to the attention maps shown in Figure 4 (left) in the main text.

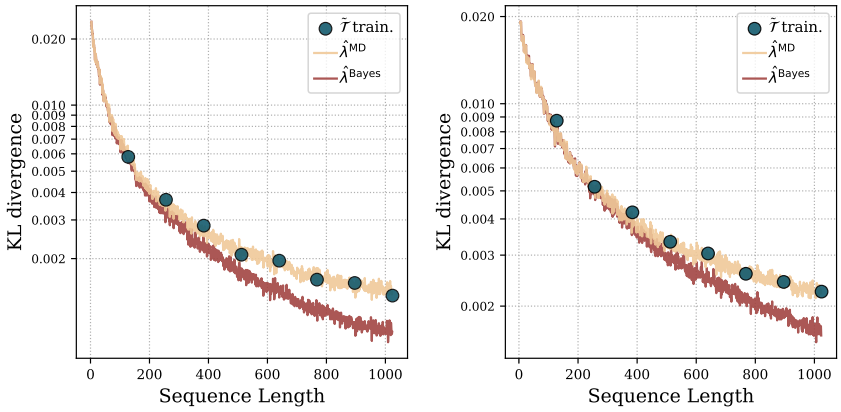


Figure 6: **KL divergence to the ground-truth** We report the KL divergence to the ground truth transition probabilities for the trained transformers, the constructed transformer, and the one-step MD estimator across sequence lengths **Left:** order 3 **Right:** order 5.

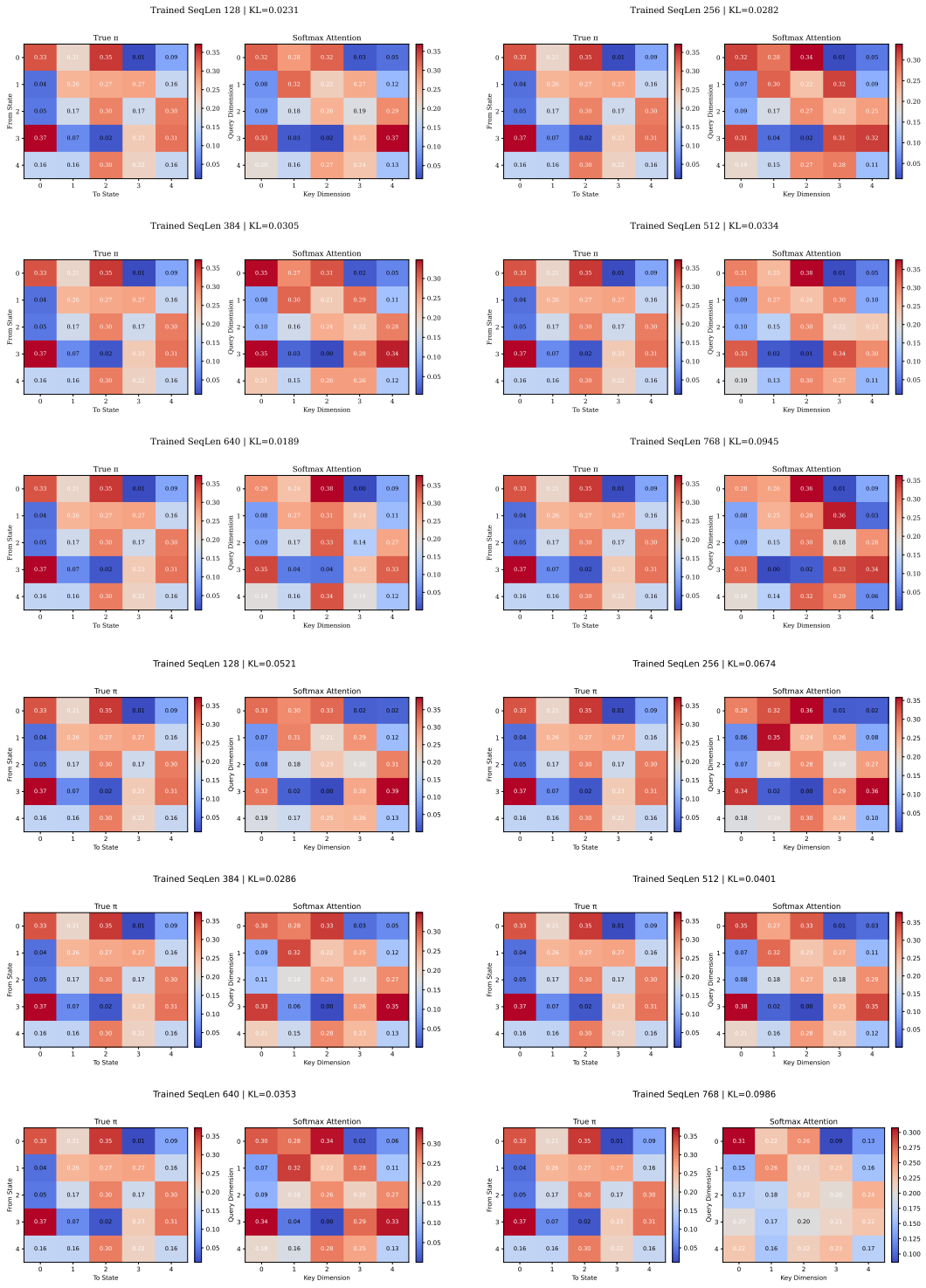


Figure 7: **Layer-1 Attention Softmax vs. True Transition matrices for orders 3 and 5.** For both orders ($m = 3$ top 3 rows, $m = 5$ bottom 3 rows), each panel reports the learned first-layer attention with the softmax applied row-wise alongside the true transition matrix; the average row-wise KL divergence is reported in the panel title. Sequence lengths increase left-to-right, top-to-bottom.

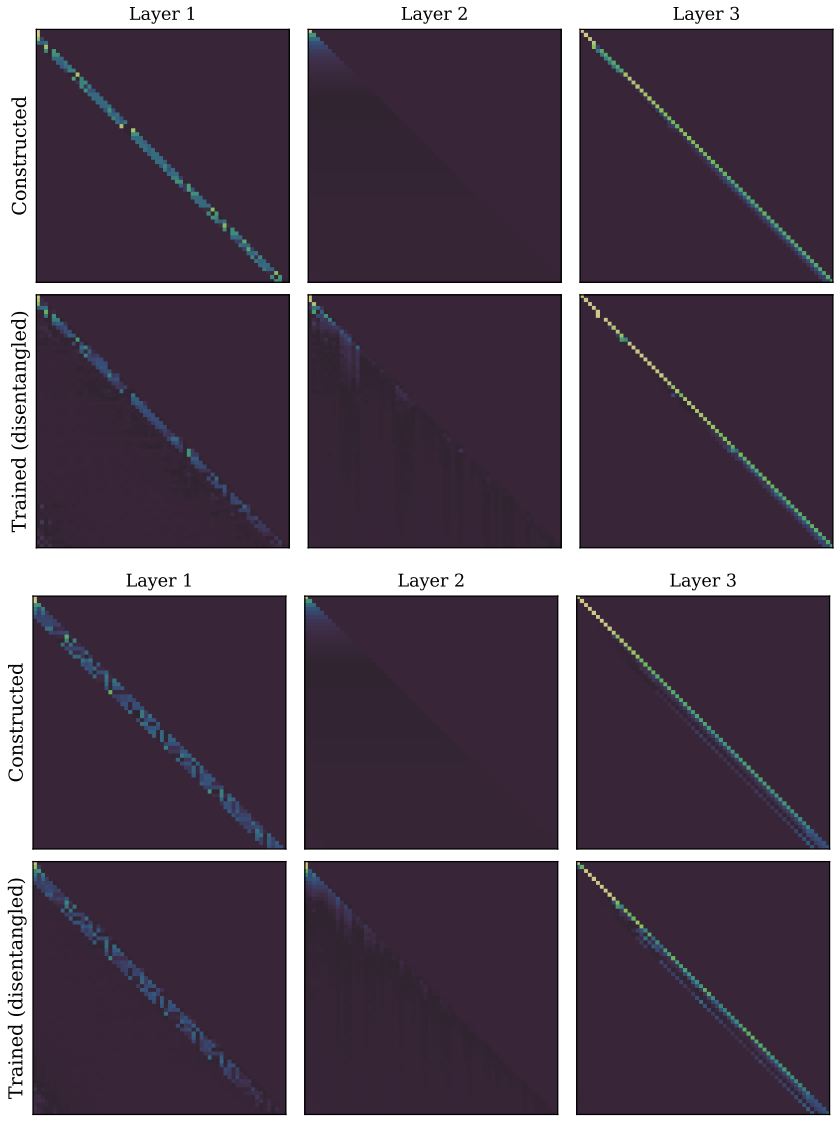


Figure 8: **Comparison of Trained and Constructed Transformers (attention grids). Top:** Attention maps of the trained transformer (disentangled) versus our theoretical construction (seq. length 64, MTD order $m = 3$). **Bottom:** Same as top but for MTD order $m = 5$.

# E Derivation of the Bayes-Optimal MTD Predictor

Here, we provide the full derivation for the Bayes-optimal predictive distribution stated in Proposition 1.

Our goal is to derive the predictive distribution for the next state, $p(Y_{t+1}|\boldsymbol{y}_1^t, \boldsymbol{\alpha})$, given an observed data prefix $\boldsymbol{y}_1^t = (y_1, \ldots, y_t)$. The unknown mixture weights $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ are assumed to be drawn from a Dirichlet prior distribution:

$$p(\boldsymbol{\lambda}|\boldsymbol{\alpha}) = \text{Dirichlet}(\boldsymbol{\lambda}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{g=1}^m \alpha_g)}{\prod_{g=1}^m \Gamma(\alpha_g)} \prod_{g=1}^m \lambda_g^{\alpha_g - 1}.$$

The likelihood of the observed data given the parameters $\boldsymbol{\lambda}$ is defined by the MTD model:

$$p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda}) = \prod_{k=m+1}^{t} p(y_k \mid \boldsymbol{y}_1^{k-1}, \boldsymbol{\lambda}) = \prod_{k=m+1}^{t} \left( \sum_{h=1}^{m} \lambda_h \pi(y_{k-h}, y_k) \right).$$

Combining the likelihood and prior via Bayes' theorem yields the posterior distribution over the mixture weights:

$$p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) \propto p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda}) \cdot p(\boldsymbol{\lambda} \mid \boldsymbol{\alpha}).$$

The Bayesian predictive distribution is formulated by marginalizing the single-step prediction $p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\lambda})$ over this posterior distribution of $\boldsymbol{\lambda}$:

$$p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) = \int_{\Delta_{m-1}} p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\lambda}) \cdot p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})\, d\boldsymbol{\lambda},$$

where $\mathcal{S}$ is the probability simplex. The single-step prediction is simply the MTD model definition:

$$p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\lambda}) = \sum_{g=1}^{m} \lambda_g \pi(y_{t+1-g}, j).$$

Substituting this into the integral gives:

$$p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) = \int_{\Delta_{m-1}} \left( \sum_{g=1}^{m} \lambda_g \pi(y_{t+1-g}, j) \right) p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})\, d\boldsymbol{\lambda}.$$

By the linearity of expectation (and integration), the integral and the finite sum can be interchanged:

$$p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) = \sum_{g=1}^{m} \pi(y_{t+1-g}, j) \left( \int_{\Delta_{m-1}} \lambda_g \cdot p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})\, d\boldsymbol{\lambda} \right).$$

We recognize the term in the parentheses as the definition of the posterior mean of the parameter $\lambda_g$:

$$\hat{\lambda}_g^{\text{Bayes}} \coloneqq \mathbb{E}[\lambda_g \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}] = \int_{\Delta_{m-1}} \lambda_g \cdot p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})\, d\boldsymbol{\lambda}.$$

This substitution yields the final form of the Bayes-optimal predictor, completing the proof:

$$p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) = \sum_{g=1}^{m} \hat{\lambda}_g^{\text{Bayes}} \cdot \pi(y_{t+1-g}, j).$$

## E.1 The Structure of the Bayes-Optimal Estimator

While the posterior mean is intractable to compute, its structure can be derived exactly. In particular it can be shown that the MTD posterior is a finite mixture of Dirichlet distributions, and from this, we can derive that the mean of the posterior preserves the classic 'add-constant' structure of conjugate Bayesian models, where the unobserved data counts are replaced by their posterior expectation.

**Proposition 4** (Posterior as a Mixture of Dirichlets)**.** *Given the MTD likelihood $L(\boldsymbol{\lambda}) = p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda})$ and a Dirichlet prior $p(\boldsymbol{\lambda} \mid \boldsymbol{\alpha}) = \mathit{Dir}(\boldsymbol{\lambda} \mid \boldsymbol{\alpha})$, the posterior distribution is a finite mixture of Dirichlet distributions:*

$$p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) = \sum_{z \in \{1,\dots,m\}^{t-m}} \pi(z) \cdot \mathit{Dir}(\boldsymbol{\lambda} \mid \boldsymbol{\alpha} + k(z)), \tag{9}$$

*where $z = (z_{m+1}, \dots, z_t)$ is a latent assignment path, $k(z)$ is the vector of counts of each lag in path $\boldsymbol{z}$, and $\pi(z) = p(Z = z \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})$ are the true posterior probabilities of the latent paths.*

*Proof.* We begin with Bayes' theorem for the posterior distribution:

$$p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) \propto p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda}) \cdot p(\boldsymbol{\lambda} \mid \boldsymbol{\alpha}).$$

The central idea is to express the observed-data likelihood, $p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda})$, by marginalizing over all possible latent assignment paths $\boldsymbol{z}$. A path $\boldsymbol{z} = (z_{m+1}, \dots, z_t)$ specifies which lag was used at each step $k$.

$$p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda}) = \sum_{\boldsymbol{z} \in \{1,\dots,m\}^{t-m}} p(\boldsymbol{y}_1^t, \boldsymbol{z} \mid \boldsymbol{\lambda}).$$

The joint probability of the data and a specific path $\boldsymbol{z}$, known as the complete-data likelihood, is given by:

$$\begin{aligned} p(\boldsymbol{y}_1^t, z \mid \boldsymbol{\lambda}) &= \prod_{k=m+1}^{t} p(y_k, z_k \mid \boldsymbol{y}_1^{k-1}, \boldsymbol{\lambda}) \\ &= \prod_{k=m+1}^{t} p(z_k \mid \boldsymbol{\lambda}) \cdot p(y_k \mid \boldsymbol{y}_1^{k-1}, z_k, \boldsymbol{\lambda}) \\ &= \prod_{k=m+1}^{t} \lambda_{z_k} \cdot \pi(y_{k-z_k}, y_k). \end{aligned}$$

We can group the terms that depend on $\boldsymbol{\lambda}$ and those that do not. Let $k_g(z) = \sum_{k=m+1}^{t} \mathbb{I}(z_k = g)$ be the number of times lag $g$ is used in path $z$. Then:

$$p(\boldsymbol{y}_1^t, z \mid \boldsymbol{\lambda}) = \left( \prod_{g=1}^{m} \lambda_g^{k_g(z)} \right) \left( \prod_{k=m+1}^{t} \pi(y_{k-z_k}, y_k) \right).$$

Let $P(\boldsymbol{y} \mid z) \coloneqq \prod_{k=m+1}^{t} \pi(y_{k-z_k}, y_k)$, which is constant with respect to $\boldsymbol{\lambda}$. The prior is given by $p(\boldsymbol{\lambda} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{g=1}^{m} \lambda_g^{\alpha_g - 1}$, where $B(\boldsymbol{\alpha})$ is the multivariate beta function. Substituting these into the expression for the posterior:

$$\begin{aligned} p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) &\propto \left( \sum_z P(\boldsymbol{y} \mid z) \prod_{g=1}^{m} \lambda_g^{k_g(z)} \right) \left( \frac{1}{B(\boldsymbol{\alpha})} \prod_{g=1}^{m} \lambda_g^{\alpha_g - 1} \right) \\ &\propto \sum_z P(\boldsymbol{y} \mid z) \prod_{g=1}^{m} \lambda_g^{\alpha_g + k_g(z) - 1}. \end{aligned}$$

We recognize that the term $\prod_g \lambda_g^{(\alpha_g + k_g(z)) - 1}$ is the kernel of a Dirichlet distribution, $\mathrm{Dir}(\boldsymbol{\lambda} \mid \boldsymbol{\alpha} + k(z))$. We can write it as $B(\boldsymbol{\alpha} + k(z)) \cdot \mathrm{Dir}(\boldsymbol{\lambda} \mid \boldsymbol{\alpha} + k(z))$. Thus, the posterior takes the form of a weighted sum of Dirichlet densities:

$$p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}) = \sum_z \pi(z) \cdot \mathrm{Dir}(\boldsymbol{\lambda} \mid \boldsymbol{\alpha} + k(z)),$$

where the mixture weights $\pi(z)$ are the normalized coefficients, which are precisely the true posterior probabilities of the latent paths, $p(Z = z \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})$. This completes the proof. □

From this mixture structure, we can derive an exact identity for the posterior mean.

**Proposition 5** (Bayes Mean as Add-Constant-to-Expected-Counts)**.** *The components of the Bayesian posterior mean, $\hat{\boldsymbol{\lambda}}^{Bayes} = \mathbb{E}[\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha}]$, are given by:*

$$\hat{\lambda}_g^{Bayes} = \frac{\alpha_g + \mathbb{E}_{Z\sim\pi}[k_g(Z)]}{\alpha_0 + (t-m)}, \tag{10}$$

*where $\alpha_0 = \sum_j \alpha_j$, and $\mathbb{E}_{Z\sim\pi}[k_g(Z)]$ is the posterior expected number of times lag $g$ was used, which is computationally intractable.*

*Proof.* The posterior mean is defined by the integral of $\boldsymbol{\lambda}$ over its posterior distribution:

$$\hat{\boldsymbol{\lambda}}^{\text{Bayes}} = \int_{\Delta_{m-1}} \boldsymbol{\lambda} \cdot p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})\, d\boldsymbol{\lambda}.$$

We substitute the mixture-of-Dirichlets form of the posterior from Proposition 4:

$$\hat{\boldsymbol{\lambda}}^{\text{Bayes}} = \int_{\Delta_{m-1}} \boldsymbol{\lambda} \cdot \left( \sum_z \pi(z) \cdot \text{Dir}(\boldsymbol{\lambda} \mid \boldsymbol{\alpha} + k(z)) \right) d\boldsymbol{\lambda}.$$

Since the summation is over a finite set of paths $z$, we can swap the integral and the summation:

$$\hat{\boldsymbol{\lambda}}^{\text{Bayes}} = \sum_z \pi(z) \left( \int_{\Delta_{m-1}} \boldsymbol{\lambda} \cdot \text{Dir}(\boldsymbol{\lambda} \mid \boldsymbol{\alpha} + k(z))\, d\boldsymbol{\lambda} \right).$$

The term inside the parentheses is the definition of the mean of a Dirichlet distribution with parameter vector $\boldsymbol{\beta} = \boldsymbol{\alpha} + k(z)$. The mean of a $\text{Dir}(\boldsymbol{\beta})$ distribution is the vector $\boldsymbol{\beta}/\beta_0$, where $\beta_0 = \sum_j \beta_j$. In our case, the sum of the parameters is:

$$\sum_{g=1}^{m} (\alpha_g + k_g(z)) = \left( \sum_g \alpha_g \right) + \left( \sum_g k_g(z) \right) = \alpha_0 + (t-m).$$

Therefore, the inner integral evaluates to the vector $\frac{\boldsymbol{\alpha}+k(z)}{\alpha_0+(t-m)}$. Substituting this back:

$$\hat{\boldsymbol{\lambda}}^{\text{Bayes}} = \sum_z \pi(z) \frac{\boldsymbol{\alpha} + k(z)}{\alpha_0 + (t-m)}.$$

This expression is an expectation over the posterior distribution of latent paths, $Z \sim \pi(z)$. We can write it as:

$$\hat{\boldsymbol{\lambda}}^{\text{Bayes}} = \mathbb{E}_{Z\sim\pi} \left[ \frac{\boldsymbol{\alpha} + k(Z)}{\alpha_0 + (t-m)} \right].$$

By the linearity of expectation, we can take the expectation inside for each component $g$:

$$\hat{\lambda}_g^{\text{Bayes}} = \frac{\mathbb{E}_{Z\sim\pi}[\alpha_g + k_g(Z)]}{\alpha_0 + (t-m)} = \frac{\alpha_g + \mathbb{E}_{Z\sim\pi}[k_g(Z)]}{\alpha_0 + (t-m)}.$$

This reveals the "add-constant-to-expected-counts" structure of the estimator, completing the proof. □

# F Approximations for the mean of the posterior distribution

In this section, we outline the methods used to approximate the mean of the posterior distribution over the mixture weights $\boldsymbol{\lambda}$.

## F.1 Markov Chain Monte Carlo (MCMC)

Since an analytical solution is unavailable, we approximate the Bayesian predictive distribution using samples from the posterior distribution generated by a Markov Chain Monte Carlo (MCMC) method, specifically **Gibbs sampling**. Gibbs sampling is well-suited for this problem because, while the full posterior is intractable, the conditional posteriors of the parameters and latent variables are simple to sample from.

The procedure involves augmenting the model with latent variables $\boldsymbol{Z}_1^t = (Z_{m+1}, \ldots, Z_t)$, where $Z_k = g$ indicates that lag $g$ was used to generate the transition to $Y_k$. The Gibbs sampler iteratively draws from the two full conditional distributions:

1. **Sample Latent Variables $Z$ given Parameters $\boldsymbol{\lambda}$:** For a given parameter vector $\boldsymbol{\lambda}^{(k-1)}$ from the previous iteration, we sample each latent variable $Z_s$ for $s \in \{m+1, \ldots, t\}$ from its categorical conditional posterior:
$$\mathbb{P}(Z_s = g \mid \boldsymbol{y}_1^t, \boldsymbol{\lambda}^{(k-1)}) = \frac{\lambda_g^{(k-1)} \pi(y_{s-g}, y_s)}{\sum_{h=1}^{m} \lambda_h^{(k-1)} \pi(y_{s-h}, y_s)}. \tag{11}$$
This provides a complete sampled sequence of lags, $\boldsymbol{z}^{(k)} = (z_{m+1}^{(k)}, \ldots, z_t^{(k)})$.
2. **Sample Parameters $\boldsymbol{\lambda}$ given Latent Variables $Z$:** Given the sampled lags $\boldsymbol{z}^{(k)}$, the Dirichlet prior is now conjugate to the complete-data likelihood. We first count the occurrences of each lag, $n_g = \sum_{s=m+1}^{t} \mathbb{I}(z_s^{(k)} = g)$. The full conditional posterior for $\boldsymbol{\lambda}$ is a Dirichlet distribution, from which we draw the next sample $\boldsymbol{\lambda}^{(k)}$:
$$p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{z}^{(k)}, \boldsymbol{\alpha}) = \text{Dirichlet}(\boldsymbol{\lambda} \mid \alpha_1 + n_1, \ldots, \alpha_m + n_m). \tag{12}$$

After running the sampler for $K_{\text{total}}$ iterations and discarding an initial burn-in period, we obtain a set of $K$ samples, $\{\boldsymbol{\lambda}^{(1)}, \boldsymbol{\lambda}^{(2)}, \ldots, \boldsymbol{\lambda}^{(K)}\}$, that are approximately drawn from the true posterior $p(\boldsymbol{\lambda} \mid \boldsymbol{y}_1^t, \boldsymbol{\alpha})$ shown in Equation 4.

The integral in Equation 4 is then approximated via a Monte Carlo average:

$$\begin{aligned}\hat{p}(Y_{t+1} = j \mid \boldsymbol{y}_1^t) &= \frac{1}{K} \sum_{k=1}^{K} p(Y_{t+1} = j \mid \boldsymbol{y}_1^t, \boldsymbol{\lambda}^{(k)}) \\ &= \frac{1}{K} \sum_{k=1}^{K} \left( \sum_{g=1}^{m} \lambda_g^{(k)} \pi(y_{t+1-g}, j) \right). \end{aligned} \tag{13}$$

This estimate converges to the true Bayes optimal predictive distribution as $K \to \infty$. It represents the theoretical performance limit for inference under the MTD model assumptions, providing a gold-standard benchmark against which other estimators can be compared.

# G Algorithms for Maximum Likelihood Estimation of MTD

Maximum Likelihood Estimation (MLE) for the Mixture Transition Distribution (MTD) does not admit a close-form solution therefore iterative optimization algorithms are required. This section details two iterative algorithms suited for this task. We first review the Expectation-Maximization (EM) algorithm, a standard and widely-used method for latent variable models. We then present Mirror Descent (MD), an alternative optimization framework that is central to our work.

## G.1 Expectation-Maximization (EM) Algorithm

The Expectation-Maximization (EM) algorithm is a widely-used iterative method for finding Maximum Likelihood estimates in statistical models with latent variables. In the context of the MTD model, the latent variables correspond to the specific mixture component responsible for generating each observation. The algorithm alternates between two steps: the Expectation (E) step, where it computes the expected log-likelihood with respect to the posterior distribution of the latent variables, and the Maximization (M) step, where it updates the model parameters to maximize this expected value.

Given the latent variables $\boldsymbol{Z} = (Z_{m+1}, \ldots, Z_n)$, where each $Z_t \in \{1, \ldots, m\}$. The variable $Z_t = g$ indicates that the $g^{th}$ mixture component (corresponding to lag $g$) was responsible for generating the transition to $Y_t$ at time $t$. The complete data are $(\boldsymbol{y}, \boldsymbol{z})$.

The likelihood of the complete data $(\boldsymbol{y}_{m+1}^n, \boldsymbol{z}_{m+1}^n)$, conditional on $\boldsymbol{y}_1^m$, is given by:

$$\begin{aligned}\mathbb{P}(\boldsymbol{y}_{m+1}^n, \boldsymbol{z}_{m+1}^n \mid \boldsymbol{y}_1^m; \boldsymbol{\lambda}) &= \prod_{t=m+1}^{n} \mathbb{P}(y_t, z_t \mid \boldsymbol{y}_1^{t-1}; \boldsymbol{\lambda}) \\ &= \prod_{t=m+1}^{n} \mathbb{P}(Z_t = z_t \mid \boldsymbol{y}_1^{t-1}; \boldsymbol{\lambda}) \mathbb{P}(y_t \mid Z_t = z_t, \boldsymbol{y}_1^{t-1}; \boldsymbol{\lambda}).\end{aligned}$$

Under the MTD model assumptions:

- $\mathbb{P}(Z_t = g \mid \boldsymbol{y}_1^{t-1}; \boldsymbol{\lambda}) = \lambda_g$
- $\mathbb{P}(y_t \mid Z_t = g, \boldsymbol{y}_1^{t-1}; \boldsymbol{\lambda}) = \pi(y_{t-g}, y_t)$

Thus, $\mathbb{P}(y_t, Z_t = g \mid \boldsymbol{y}_1^{t-1}; \boldsymbol{\lambda}) = \lambda_g \pi(y_{t-g}, y_t)$. The complete data likelihood becomes:

$$\mathbb{P}(\boldsymbol{y}_{m+1}^n, \boldsymbol{z}_{m+1}^n \mid \boldsymbol{y}_1^m; \boldsymbol{\lambda}) = \prod_{t=m+1}^{n} \lambda_{z_t} \pi_{z_t}(y_{t-z_t}, y_t).$$

The complete data log-likelihood, $\ell_c(\boldsymbol{\lambda}; \boldsymbol{y}, \boldsymbol{z}) = \log \mathbb{P}(\boldsymbol{y}_{m+1}^n, \boldsymbol{z}_{m+1}^n \mid \boldsymbol{y}_1^m; \boldsymbol{\lambda})$, is:

$$\begin{aligned} \ell_c(\boldsymbol{\lambda}; \boldsymbol{y}, \boldsymbol{z}) &= \sum_{t=m+1}^{n} \log(\lambda_{z_t} \pi_{z_t}(y_{t-z_t}, y_t)) \\ &= \sum_{t=m+1}^{n} \sum_{g=1}^{m} \mathbb{I}(z_t = g) \log(\lambda_g \pi(y_{t-g}, y_t)), \end{aligned} \tag{14}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

### G.2 EM Algorithm Steps

Let $\boldsymbol{\lambda}^{(k)}$ be the estimate of $\boldsymbol{\lambda}$ at iteration $k$.

**E-Step (Expectation)** The E-step computes the expectation of the complete-data log-likelihood equation 14 with respect to the conditional distribution of the latent variables $\boldsymbol{Z}$ given the observed data $\boldsymbol{y}$ and the current parameter estimate $\boldsymbol{\lambda}^{(k)}$. This expectation defines the $Q$ function:

$$\begin{aligned} Q(\boldsymbol{\lambda} \mid \boldsymbol{\lambda}^{(k)}) &= \mathbb{E}_{\boldsymbol{Z} \mid \boldsymbol{y}, \boldsymbol{\lambda}^{(k)}}[\ell_c(\boldsymbol{\lambda}; \boldsymbol{y}, \boldsymbol{Z})] \\ &= \mathbb{E}\left[ \sum_{t=m+1}^{n} \sum_{g=1}^{m} \mathbb{I}(Z_t = g) \log(\lambda_g \pi(y_{t-g}, y_t)) \;\middle|\; \boldsymbol{y}, \boldsymbol{\lambda}^{(k)} \right] \\ &= \sum_{t=m+1}^{n} \sum_{g=1}^{m} \mathbb{E}[\mathbb{I}(Z_t = g) \mid \boldsymbol{y}, \boldsymbol{\lambda}^{(k)}] \log(\lambda_g \pi(y_{t-g}, y_t)). \end{aligned}$$

The core computation is the posterior probability (responsibility) of $Z_t = g$:

$$\gamma_t^{(k)}(g) := \mathbb{E}[\mathbb{I}(Z_t = g) \mid \boldsymbol{y}, \boldsymbol{\lambda}^{(k)}] = \mathbb{P}(Z_t = g \mid \boldsymbol{y}, \boldsymbol{\lambda}^{(k)}).$$

Due to the MTD model structure, future observations $\boldsymbol{y}_{t+1}^n$ are conditionally independent of $Z_t$ given $\boldsymbol{y}_1^t$. Thus, the posterior probability simplifies:

$$\mathbb{P}(Z_t = g \mid \boldsymbol{y}, \boldsymbol{\lambda}^{(k)}) = \mathbb{P}(Z_t = g \mid \boldsymbol{y}_1^t, \boldsymbol{\lambda}^{(k)}).$$

Using Bayes' theorem:

$$\begin{aligned} \gamma_t^{(k)}(g) &= \mathbb{P}(Z_t = g \mid \boldsymbol{y}_1^t, \boldsymbol{\lambda}^{(k)}) \\ &= \frac{\mathbb{P}(y_t \mid Z_t = g, \boldsymbol{y}_1^{t-1}, \boldsymbol{\lambda}^{(k)}) \mathbb{P}(Z_t = g \mid \boldsymbol{y}_1^{t-1}, \boldsymbol{\lambda}^{(k)})}{\mathbb{P}(y_t \mid \boldsymbol{y}_1^{t-1}, \boldsymbol{\lambda}^{(k)})} \\ &= \frac{\pi(y_{t-g}, y_t) \lambda_g^{(k)}}{\sum_{h=1}^{m} \mathbb{P}(y_t, Z_t = h \mid \boldsymbol{y}_1^{t-1}, \boldsymbol{\lambda}^{(k)})} \\ &= \frac{\lambda_g^{(k)} \pi(y_{t-g}, y_t)}{\sum_{h=1}^{m} \lambda_h^{(k)} \pi_h(y_{t-h}, y_t)}. \end{aligned} \tag{15}$$

The E-step involves calculating these responsibilities $\gamma_t^{(k)}(g)$ for all $t \in \{m+1, \ldots, n\}$ and $g \in \{1, \ldots, m\}$. The $Q$ function is then:

$$Q(\boldsymbol{\lambda} \mid \boldsymbol{\lambda}^{(k)}) = \sum_{t=m+1}^{n} \sum_{g=1}^{m} \gamma_t^{(k)}(g)(\log \lambda_g + \log \pi(y_{t-g}, y_t)). \tag{16}$$

**M-Step (Maximization)** The M-step finds the parameter values $\boldsymbol{\lambda}$ that maximize the $Q$ function equation 16 subject to the constraints $\lambda_g \geq 0$ and $\sum_{g=1}^{m} \lambda_g = 1$. This gives the updated estimate $\boldsymbol{\lambda}^{(k+1)}$.

$$\boldsymbol{\lambda}^{(k+1)} = \arg\max_{\boldsymbol{\lambda}} Q(\boldsymbol{\lambda} \mid \boldsymbol{\lambda}^{(k)}).$$

Since the terms $\log \pi(y_{t-g}, y_t)$ do not depend on $\boldsymbol{\lambda}$, we maximize:

$$f(\boldsymbol{\lambda}) = \sum_{t=m+1}^{n} \sum_{g=1}^{m} \gamma_t^{(k)}(g) \log \lambda_g = \sum_{g=1}^{m} \left( \sum_{t=m+1}^{n} \gamma_t^{(k)}(g) \right) \log \lambda_g.$$

Let $C_g = \sum_{t=m+1}^{n} \gamma_t^{(k)}(g)$. We maximize $f(\boldsymbol{\lambda}) = \sum_{g=1}^{m} C_g \log \lambda_g$ subject to $\sum_{g=1}^{m} \lambda_g = 1$. We use a Lagrange multiplier $\mu$:

$$\mathcal{L}(\boldsymbol{\lambda}, \mu) = \sum_{g=1}^{m} C_g \log \lambda_g - \mu \left( \sum_{g=1}^{m} \lambda_g - 1 \right).$$

Setting partial derivatives to zero:

$$\frac{\partial \mathcal{L}}{\partial \lambda_g} = \frac{C_g}{\lambda_g} - \mu = 0 \implies \lambda_g = \frac{C_g}{\mu}$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = -\left( \sum_{g=1}^{m} \lambda_g - 1 \right) = 0 \implies \sum_{g=1}^{m} \lambda_g = 1.$$

Substituting $\lambda_g = C_g/\mu$ into the constraint yields $\mu = \sum_{h=1}^{m} C_h$. Therefore:

$$\lambda_g = \frac{C_g}{\sum_{h=1}^{m} C_h} = \frac{\sum_{t=m+1}^{n} \gamma_t^{(k)}(g)}{\sum_{h=1}^{m} \sum_{t'=m+1}^{n} \gamma_{t'}^{(k)}(h)}.$$

The denominator simplifies as $\sum_{h=1}^{m} \sum_{t'=m+1}^{n} \gamma_{t'}^{(k)}(h) = \sum_{t'=m+1}^{n} \sum_{h=1}^{m} \gamma_{t'}^{(k)}(h) = \sum_{t'=m+1}^{n} 1 = n - m$. The M-step update rule is thus:

$$\lambda_g^{(k+1)} = \frac{\sum_{t=m+1}^{n} \gamma_t^{(k)}(g)}{n - m}. \tag{17}$$

### G.3 Summary of the EM Algorithm

The EM algorithm for estimating the mixture weights $\boldsymbol{\lambda}$ in the MTD model, assuming known transition matrices $\boldsymbol{\pi}_g$, proceeds as follows:

1. **Initialization:** Choose initial weights $\boldsymbol{\lambda}^{(0)} = (\lambda_1^{(0)}, \ldots, \lambda_m^{(0)})$ such that $\lambda_g^{(0)} \geq 0$ for all $g \in \{1, \ldots, m\}$ and $\sum_{g=1}^{m} \lambda_g^{(0)} = 1$. Set the iteration counter $k = 0$.
2. **E-Step (Expectation):** Compute the responsibilities $\gamma_t^{(k)}(g)$ for each time point $t \in \{m+1, \ldots, n\}$ and each mixture component $g \in \{1, \ldots, m\}$, using the current parameter estimates $\boldsymbol{\lambda}^{(k)}$:
$$\gamma_t^{(k)}(g) = \frac{\lambda_g^{(k)} \pi(y_{t-g}, y_t)}{\sum_{h=1}^{m} \lambda_h^{(k)} \pi_h(y_{t-h}, y_t)}. \tag{18}$$
3. **M-Step (Maximization):** Update the mixture weights $\lambda_g^{(k+1)}$ for each component $g \in \{1, \ldots, m\}$ using the computed responsibilities:
$$\lambda_g^{(k+1)} = \frac{\sum_{t=m+1}^{n} \gamma_t^{(k)}(g)}{n - m}. \tag{19}$$
4. **Convergence Check:** If the change in the parameter estimates (e.g., $||\boldsymbol{\lambda}^{(k+1)} - \boldsymbol{\lambda}^{(k)}||$) or the change in the observed data log-likelihood (e.g., $\ell(\boldsymbol{\lambda}^{(k+1)}; \boldsymbol{y}) - \ell(\boldsymbol{\lambda}^{(k)}; \boldsymbol{y})$, where $\ell(\boldsymbol{\lambda}; \boldsymbol{y})$ is the observed data log-likelihood) is below a predefined tolerance $\epsilon$, stop the algorithm and return $\hat{\boldsymbol{\lambda}} = \boldsymbol{\lambda}^{(k+1)}$ as the estimated mixture weights. Otherwise, set $k \leftarrow k + 1$ and repeat from Step 2.

### G.4 Mirror Descent

This appendix provides detailed derivations for Equation (5) and Proposition 2.

### G.5 Derivation of the Exponentiated Gradient Algorithm

The Exponentiated Gradient (EG) algorithm is a specific instance of the Mirror Descent (MD) online optimization algorithm. We apply it to the problem of maximizing the MTD log-likelihood function, $\ell(\boldsymbol{\lambda})$, with respect to the mixture weights $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$. The weights are constrained to the probability simplex, $\Delta_{m-1} = \{\boldsymbol{\lambda} \in \mathbb{R}^m \mid \sum_g \lambda_g = 1, \lambda_g \geq 0\}$.

The general MD update step at iteration $k$ linearizes the objective function around the current estimate $\boldsymbol{\lambda}^{(k)}$ and adds a regularization term. The next iterate, $\boldsymbol{\lambda}^{(k+1)}$, is found by solving:

$$\boldsymbol{\lambda}^{(k+1)} = \underset{\boldsymbol{\lambda} \in \Delta_{m-1}}{\arg\max} \left\{ \langle \nabla \ell(\boldsymbol{\lambda}^{(k)}), \boldsymbol{\lambda} \rangle - \frac{1}{\eta} D_\Psi(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(k)}) \right\}, \tag{20}$$

where $\eta > 0$ is the learning rate, $\nabla \ell(\boldsymbol{\lambda}^{(k)})$ is the gradient of the log-likelihood evaluated at $\boldsymbol{\lambda}^{(k)}$, and $D_\Psi$ is the Bregman divergence associated with a potential function $\Psi$. For optimization over the simplex, the standard choice is the negative entropy potential, $\Psi(\boldsymbol{\lambda}) = \sum_{g=1}^m \lambda_g \log \lambda_g$. The resulting Bregman divergence is the unnormalized Kullback-Leibler (KL) divergence:

$$D_\Psi(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(k)}) = \sum_{g=1}^m \lambda_g \log \frac{\lambda_g}{\lambda_g^{(k)}}. \tag{21}$$

Therefore the optizazion problem in Equation (20) becomes:

$$\boldsymbol{\lambda}^{(k+1)} = \underset{\boldsymbol{\lambda} \in \Delta_{m-1}}{\arg\max} \left\{ \langle \nabla \ell(\boldsymbol{\lambda}^{(k)}), \boldsymbol{\lambda} \rangle - \frac{1}{\eta} \sum_{g=1}^m \lambda_g \log \frac{\lambda_g}{\lambda_g^{(k)}} \right\}. \tag{22}$$

To solve the optimization problem in Eq. 20, we form the Lagrangian with a multiplier $\mu$ for the constraint $\sum_g \lambda_g = 1$:

$$\mathcal{L}(\boldsymbol{\lambda}, \mu) = \eta \langle \nabla \ell(\boldsymbol{\lambda}^{(k)}), \boldsymbol{\lambda} \rangle - \sum_g \lambda_g \log \frac{\lambda_g}{\lambda_g^{(k)}} - \mu \left( \sum_g \lambda_g - 1 \right). \tag{23}$$

Setting the derivative $\partial \mathcal{L} / \partial \lambda_g$ to zero yields:

$$\begin{aligned} \eta \nabla_\lambda \ell(\boldsymbol{\lambda}^{(k)})_g - \left( \log \frac{\lambda_g}{\lambda_g^{(k)}} + 1 \right) - \mu = 0 \\ \log \frac{\lambda_g}{\lambda_g^{(k)}} = \eta \nabla_\lambda \ell(\boldsymbol{\lambda}^{(k)})_g - \mu - 1 \\ \lambda_g = \lambda_g^{(k)} \exp(\eta \nabla_\lambda \ell(\boldsymbol{\lambda}^{(k)})_g) \exp(-\mu - 1). \end{aligned}$$

The term $\exp(-\mu - 1)$ serves as a normalization constant to ensure $\sum_g \lambda_g = 1$. This leads directly to the EG update rule presented in Equation (5):

$$\lambda_g^{(k+1)} = \frac{\lambda_g^{(k)} \exp(\eta \cdot \nabla_\lambda \ell(\boldsymbol{\lambda}^{(k)})_g)}{\sum_{h=1}^m \lambda_h^{(k)} \exp(\eta \cdot \nabla_\lambda \ell(\boldsymbol{\lambda}^{(k)})_h)}. \tag{5, repeated}$$

### G.6 Derivation of the One-Step MTD Estimator

We now prove Proposition 2 by specializing the EG update to the MTD model and evaluating it at the uniform prior $\boldsymbol{\lambda}^{(0)} = (1/m, \ldots, 1/m)$.

**Step 1: MTD Log-Likelihood and its Gradient.** Given an observed sequence prefix $\boldsymbol{y}_1^t$, the MTD log-likelihood is:

$$\ell(\boldsymbol{\lambda}) = \log p(\boldsymbol{y}_1^t \mid \boldsymbol{\lambda}) = \sum_{k=m+1}^{t} \log \left( \sum_{h=1}^{m} \lambda_h \pi(y_{k-h}, y_k) \right). \tag{24}$$

The $g$-th component of its gradient is:

$$\nabla_\lambda \ell(\boldsymbol{\lambda})_g = \frac{\partial \ell(\boldsymbol{\lambda})}{\partial \lambda_g} = \sum_{k=m+1}^{t} \frac{\pi(y_{k-g}, y_k)}{\sum_{h=1}^{m} \lambda_h \pi(y_{k-h}, y_k)}. \tag{25}$$

**Step 2: Evaluate Gradient at the Uniform Prior.** We evaluate this gradient at $\boldsymbol{\lambda}^{(0)} = (1/m, \ldots, 1/m)$:

$$\begin{aligned} \nabla_\lambda \ell(\boldsymbol{\lambda}^{(0)})_g &= \sum_{k=m+1}^{t} \frac{\pi(y_{k-g}, y_k)}{\sum_{h=1}^{m} (1/m) \pi(y_{k-h}, y_k)} \\ &= m \sum_{k=m+1}^{t} \frac{\pi(y_{k-g}, y_k)}{\sum_{h=1}^{m} \pi(y_{k-h}, y_k)}. \end{aligned} \tag{26}$$

We recognize the term inside the summation as the posterior responsibility of lag $g$ under the uniform model:

$$\gamma_k(g) := p(Z_k = g \mid \boldsymbol{y}_1^k, \boldsymbol{\lambda}^{(0)}) \tag{27}$$

$$= \frac{p(y_k \mid y_{k-g}) p(Z_k = g | \boldsymbol{\lambda}^{(0)})}{\sum_h p(y_k \mid y_{k-h}) p(Z_k = h | \boldsymbol{\lambda}^{(0)})} \tag{28}$$

$$= \frac{\pi(y_{k-g}, y_k) \cdot (1/m)}{\sum_h \pi(y_{k-h}, y_k) \cdot (1/m)} \tag{29}$$

$$= \frac{\pi(y_{k-g}, y_k)}{\sum_{h=1}^{m} \pi(y_{k-h}, y_k)}. \tag{30}$$

Thus, the gradient at the uniform prior is a scaled sum of these responsibilities:

$$\nabla_\lambda \ell(\boldsymbol{\lambda}^{(0)})_g = m \sum_{k=m+1}^{t} \gamma_k^{\text{unif}}(g). \tag{31}$$

**Step 3: Apply the EG Update Rule.** Finally, we substitute $\lambda_g^{(0)} = 1/m$ and the derived gradient into the EG update rule (Eq. 5) to find $\lambda_g^{(1)}$:

$$\begin{aligned} \hat{\boldsymbol{\lambda}}_g^{\text{MD}} = \lambda_g^{(1)} &= \frac{\lambda_g^{(0)} \exp\left(\eta \cdot \nabla_\lambda \ell(\boldsymbol{\lambda}^{(0)})_g\right)}{\sum_{j=1}^{m} \lambda_j^{(0)} \exp\left(\eta \cdot \nabla_\lambda \ell(\boldsymbol{\lambda}^{(0)})_j\right)} \\ &= \frac{(1/m) \cdot \exp\left(\eta \cdot m \sum_{k=m+1}^{t} \gamma_k^{\text{unif}}(g)\right)}{\sum_{j=1}^{m} (1/m) \cdot \exp\left(\eta \cdot m \sum_{k=m+1}^{t} \gamma_k^{\text{unif}}(j)\right)} \\ &= \frac{\exp\left(\eta m \sum_{k=m+1}^{t} \gamma_k^{\text{unif}}(g)\right)}{\sum_{j=1}^{m} \exp\left(\eta m \sum_{k=m+1}^{t} \gamma_k^{\text{unif}}(j)\right)}. \end{aligned} \tag{32}$$

This completes the proof of Proposition 2.

## H One-Step MD as a First-Order Bayesian Approximation

In this section, we formally establish a theoretical connection between the one-step Mirror Descent (MD) estimator and the true Bayesian posterior mean. We demonstrate that their respective first-order Taylor expansions around a state of "no evidence" are identical up to a scalar constant. This

result provides a rigorous basis for understanding the one-step MD estimator as a principled approximation to the Bayes-optimal predictor, particularly in a low-data or low-signal regime.

The analysis hinges on treating both estimators as functions of the log-likelihood gradient evaluated at the center of the simplex, $\boldsymbol{g} \coloneqq \nabla_{\boldsymbol{\lambda}} \ell(\boldsymbol{\lambda}^{(0)})$, and expanding them around the point of no evidence, $\boldsymbol{g} = \mathbf{0}$. The approximations will be expressed using Landau notation, where a vector function $\boldsymbol{f}(\boldsymbol{g}) = O(\|\boldsymbol{g}\|^p)$ signifies that $\|\boldsymbol{f}(\boldsymbol{g})\| \leq C\|\boldsymbol{g}\|^p$ for some constant $C$ in a neighborhood of $\boldsymbol{g} = \mathbf{0}$.

**Proposition 6** (First-Order Approximation of the One-Step MD Estimator)**.** *Let $\hat{\boldsymbol{\lambda}}^{MD}(\boldsymbol{g})$ be the one-step MD estimator defined as $\hat{\boldsymbol{\lambda}}^{MD}(\boldsymbol{g}) = \mathrm{softmax}(\eta \boldsymbol{g})$, viewed as a function of the log-likelihood gradient $\boldsymbol{g}$. Its first-order Taylor expansion around $\boldsymbol{g} = \mathbf{0}$ is given by:*

$$\hat{\lambda}_k^{MD}(\boldsymbol{g}) = \frac{1}{m} + \frac{\eta}{m}(g_k - \bar{g}) + O(\|\boldsymbol{g}\|^2), \tag{33}$$

*where $\bar{g} = \frac{1}{m}\sum_{j=1}^m g_j$. In vector form, this is $\hat{\boldsymbol{\lambda}}^{MD}(\boldsymbol{g}) = \boldsymbol{\lambda}^{(0)} + \frac{\eta}{m}\mathrm{Proj}_{\Delta}(\boldsymbol{g}) + O(\|\boldsymbol{g}\|^2)$, where $\boldsymbol{\lambda}^{(0)} = (1/m, \ldots, 1/m)$ and $\mathrm{Proj}_{\Delta}$ is the operator that projects a vector onto the hyperplane of vectors that sum to zero.*

*Proof.* The one-step MD update is given by the softmax function, $\hat{\lambda}_k^{\mathrm{MD}}(\boldsymbol{g}) = \frac{\exp(\eta g_k)}{\sum_{j=1}^m \exp(\eta g_j)}$. Since the exponential function is analytic, the softmax function is also analytic in $\boldsymbol{g}$, and its Taylor series expansion around $\boldsymbol{g} = \mathbf{0}$ exists. The expansion is given by:

$$\hat{\boldsymbol{\lambda}}^{\mathrm{MD}}(\boldsymbol{g}) = \hat{\boldsymbol{\lambda}}^{\mathrm{MD}}(\mathbf{0}) + J_{\hat{\boldsymbol{\lambda}}^{\mathrm{MD}}}(\mathbf{0})\boldsymbol{g} + O(\|\boldsymbol{g}\|^2),$$

where $J_{\hat{\boldsymbol{\lambda}}^{\mathrm{MD}}}(\mathbf{0})$ is the Jacobian matrix of $\hat{\boldsymbol{\lambda}}^{\mathrm{MD}}(\boldsymbol{g})$ evaluated at $\boldsymbol{g} = \mathbf{0}$.

**Zeroth-Order Term:** At $\boldsymbol{g} = \mathbf{0}$, the estimator evaluates to the uniform distribution, which is the prior mean $\boldsymbol{\lambda}^{(0)}$:

$$\hat{\lambda}_k^{\mathrm{MD}}(\mathbf{0}) = \frac{\exp(0)}{\sum_{j=1}^m \exp(0)} = \frac{1}{m} = \lambda_k^{(0)}.$$

**First-Order Term (Jacobian):** The entries of the Jacobian matrix, $J_{kj}(\boldsymbol{g}) = \frac{\partial}{\partial g_j}\hat{\lambda}_k^{\mathrm{MD}}(\boldsymbol{g})$, are given by $\eta \cdot \hat{\lambda}_k^{\mathrm{MD}}(\boldsymbol{g})(\delta_{kj} - \hat{\lambda}_j^{\mathrm{MD}}(\boldsymbol{g}))$. Evaluating at $\boldsymbol{g} = \mathbf{0}$, where $\hat{\lambda}_j^{\mathrm{MD}}(\mathbf{0}) = 1/m$ for all $j$:

$$\left.\frac{\partial \hat{\lambda}_k^{\mathrm{MD}}}{\partial g_j}\right|_{\boldsymbol{g}=\mathbf{0}} = \eta \cdot \frac{1}{m}\left(\delta_{kj} - \frac{1}{m}\right).$$

**Assembling the Expansion:** The $k$-th component of the expansion is $\hat{\lambda}_k^{\mathrm{MD}}(\boldsymbol{g}) = \hat{\lambda}_k^{\mathrm{MD}}(\mathbf{0}) + \sum_{j=1}^m \left.\frac{\partial \hat{\lambda}_k^{\mathrm{MD}}}{\partial g_j}\right|_{\mathbf{0}} \cdot g_j + O(\|\boldsymbol{g}\|^2)$:

$$\begin{aligned}
\hat{\lambda}_k^{\mathrm{MD}}(\boldsymbol{g}) &= \frac{1}{m} + \sum_{j=1}^m \left[\frac{\eta}{m}\left(\delta_{kj} - \frac{1}{m}\right)\right] g_j + O(\|\boldsymbol{g}\|^2) \\
&= \frac{1}{m} + \frac{\eta}{m}\left(\sum_{j=1}^m \delta_{kj} g_j - \frac{1}{m}\sum_{j=1}^m g_j\right) + O(\|\boldsymbol{g}\|^2) \\
&= \frac{1}{m} + \frac{\eta}{m}(g_k - \bar{g}) + O(\|\boldsymbol{g}\|^2).
\end{aligned}$$

This completes the proof. □

Next, we derive the corresponding approximation for the Bayesian posterior mean. We linearize the log-likelihood around the prior mean, $\boldsymbol{\lambda}^{(0)}$, which allows for an analytical treatment of the posterior.

**Proposition 7** (First-Order Approximation of the Bayesian Posterior Mean)**.** *Consider a Bayesian model with a uniform Dirichlet($\mathbf{1}$) prior over $\boldsymbol{\lambda} \in \Delta_{m-1}$ and a log-likelihood linearized around*

*the prior mean, $\ell(\boldsymbol{\lambda}) = \ell(\boldsymbol{\lambda}^{(0)}) + \langle \boldsymbol{g}, \boldsymbol{\lambda} - \boldsymbol{\lambda}^{(0)} \rangle$. The resulting posterior mean, $\hat{\boldsymbol{\lambda}}^{Bayes}(\boldsymbol{g})$, has a first-order Taylor expansion around $\boldsymbol{g} = \mathbf{0}$ given by:*

$$\hat{\lambda}_k^{Bayes}(\boldsymbol{g}) = \frac{1}{m} + \frac{1}{m(m+1)}(g_k - \bar{g}) + O(\|\boldsymbol{g}\|^2). \tag{34}$$

*In vector form, this is $\hat{\boldsymbol{\lambda}}^{Bayes}(\boldsymbol{g}) = \boldsymbol{\lambda}^{(0)} + \mathrm{Cov}_{\boldsymbol{\lambda}^{(0)}}(\boldsymbol{\lambda})\boldsymbol{g} + O(\|\boldsymbol{g}\|^2)$, where $\mathrm{Cov}_{\boldsymbol{\lambda}^{(0)}}(\boldsymbol{\lambda})$ is the covariance matrix of the prior distribution.*

*Proof.* Under the linearized likelihood, the posterior density is $p(\boldsymbol{\lambda}|\boldsymbol{g}) \propto p(\boldsymbol{\lambda}) \exp(\ell(\boldsymbol{\lambda})) \propto \exp(\langle \boldsymbol{g}, \boldsymbol{\lambda} \rangle)$, where terms constant in $\boldsymbol{\lambda}$ are absorbed into the normalization constant. The posterior mean is:

$$\hat{\boldsymbol{\lambda}}^{\text{Bayes}}(\boldsymbol{g}) = \frac{\int_{\Delta_{m-1}} \boldsymbol{\lambda} \cdot \exp(\langle \boldsymbol{g}, \boldsymbol{\lambda} \rangle)\, d\mu(\boldsymbol{\lambda})}{\int_{\Delta_{m-1}} \exp(\langle \boldsymbol{g}, \boldsymbol{\lambda} \rangle)\, d\mu(\boldsymbol{\lambda})},$$

where $d\mu(\boldsymbol{\lambda})$ is the uniform probability measure over the simplex $\Delta_{m-1}$. Let $N_k(\boldsymbol{g})$ be the numerator's $k$-th component and $Z(\boldsymbol{g})$ be the denominator. The function $\hat{\boldsymbol{\lambda}}^{\text{Bayes}}(\boldsymbol{g})$ is analytic, allowing a Taylor expansion.

**Zeroth-Order Term:** At $\boldsymbol{g} = \mathbf{0}$, the exponential term is 1. The posterior equals the prior, so the posterior mean is the prior mean:

$$\hat{\boldsymbol{\lambda}}^{\text{Bayes}}(\mathbf{0}) = \frac{\int_{\Delta_{m-1}} \boldsymbol{\lambda}\, d\mu(\boldsymbol{\lambda})}{\int_{\Delta_{m-1}} 1\, d\mu(\boldsymbol{\lambda})} = \mathbb{E}_{\boldsymbol{\lambda} \sim \text{Dir}(\mathbf{1})}[\boldsymbol{\lambda}] = \boldsymbol{\lambda}^{(0)}.$$

**First-Order Term (Jacobian):** The Jacobian entries are $\frac{\partial \hat{\lambda}_k}{\partial g_j} = \frac{\partial}{\partial g_j}\left(\frac{N_k}{Z}\right)$. Using the quotient rule:

$$\frac{\partial \hat{\lambda}_k}{\partial g_j} = \frac{1}{Z^2}\left(Z \frac{\partial N_k}{\partial g_j} - N_k \frac{\partial Z}{\partial g_j}\right).$$

We find the required derivatives of $N_k(\boldsymbol{g})$ and $Z(\boldsymbol{g})$ differentiating under the integral sign, which is applicable here as the integrands are continuous on the compact domain $\Delta_{m-1}$:

$$\frac{\partial Z}{\partial g_j} = \frac{\partial}{\partial g_j} \int_{\Delta_{m-1}} \exp\left(\sum_i g_i \lambda_i\right) d\mu(\boldsymbol{\lambda}) = \int_{\Delta_{m-1}} \lambda_j \exp\left(\sum_i g_i \lambda_i\right) d\mu(\boldsymbol{\lambda})$$

$$\frac{\partial N_k}{\partial g_j} = \frac{\partial}{\partial g_j} \int_{\Delta_{m-1}} \lambda_k \exp\left(\sum_i g_i \lambda_i\right) d\mu(\boldsymbol{\lambda}) = \int_{\Delta_{m-1}} \lambda_k \lambda_j \exp\left(\sum_i g_i \lambda_i\right) d\mu(\boldsymbol{\lambda})$$

Now, we evaluate these components at $\boldsymbol{g} = \mathbf{0}$:

- $Z(\mathbf{0}) = \int 1\, d\mu(\boldsymbol{\lambda}) = 1$.
- $N_k(\mathbf{0}) = \int \lambda_k\, d\mu(\boldsymbol{\lambda}) = \mathbb{E}[\lambda_k] = 1/m$.
- $\left.\frac{\partial Z}{\partial g_j}\right|_{\boldsymbol{g}=\mathbf{0}} = \int_{\Delta_{m-1}} \lambda_j \exp(\langle \mathbf{0}, \boldsymbol{\lambda} \rangle)\, d\mu(\boldsymbol{\lambda}) = \int \lambda_j\, d\mu(\boldsymbol{\lambda}) = \mathbb{E}[\lambda_j] = 1/m$.
- $\left.\frac{\partial N_k}{\partial g_j}\right|_{\boldsymbol{g}=\mathbf{0}} = \int_{\Delta_{m-1}} \lambda_k \lambda_j \exp(\langle \mathbf{0}, \boldsymbol{\lambda} \rangle)\, d\mu(\boldsymbol{\lambda}) = \int \lambda_k \lambda_j\, d\mu(\boldsymbol{\lambda}) = \mathbb{E}[\lambda_k \lambda_j]$.

Substituting these evaluated terms into the quotient rule expression at $\boldsymbol{g} = \mathbf{0}$:

$$\left.\frac{\partial \hat{\lambda}_k}{\partial g_j}\right|_{\boldsymbol{g}=\mathbf{0}} = \frac{1 \cdot \mathbb{E}[\lambda_k \lambda_j] - (1/m) \cdot (1/m)}{1^2} = \mathbb{E}[\lambda_k \lambda_j] - \mathbb{E}[\lambda_k]\mathbb{E}[\lambda_j] = \mathrm{Cov}_{\boldsymbol{\lambda}^{(0)}}(\lambda_k, \lambda_j).$$

This establishes that the Jacobian of the posterior mean at $\boldsymbol{g} = \mathbf{0}$ is the prior covariance matrix.

**Assembling the Expansion:** For a Dirichlet($\mathbf{1}$) distribution, the covariance matrix is $\mathrm{Cov}(\lambda_k, \lambda_j) = \frac{m\delta_{kj}-1}{m^2(m+1)}$. The $k$-th component of the expansion is:

$$\begin{aligned}\hat{\lambda}_k^{\text{Bayes}}(\boldsymbol{g}) &= \frac{1}{m} + \sum_{j=1}^{m} \frac{m\delta_{kj}-1}{m^2(m+1)} g_j + O(\|\boldsymbol{g}\|^2) \\ &= \frac{1}{m} + \frac{1}{m^2(m+1)} \left( m g_k - \sum_{j=1}^{m} g_j \right) + O(\|\boldsymbol{g}\|^2) \\ &= \frac{1}{m} + \frac{m}{m^2(m+1)} (g_k - \bar{g}) + O(\|\boldsymbol{g}\|^2) \\ &= \frac{1}{m} + \frac{1}{m(m+1)} (g_k - \bar{g}) + O(\|\boldsymbol{g}\|^2).\end{aligned}$$

This completes the proof of the proposition. □

By comparing the results of Proposition 6 and Proposition 7, we arrive at our main result.

**Theorem 3** (First-Order Equivalence of the Estimators)**.** *Let $\hat{\boldsymbol{\lambda}}^{MD}(\boldsymbol{g}; \eta)$ be the one-step MD estimator with learning rate $\eta$, and let $\hat{\boldsymbol{\lambda}}^{Bayes}(\boldsymbol{g})$ be the Bayesian posterior mean under the linearized likelihood. There exists a unique learning rate $\eta = \frac{1}{m+1}$ for which the two estimators are first-order equivalent at $\boldsymbol{g} = \mathbf{0}$.*

*Proof.* Two estimators are first-order equivalent at $\boldsymbol{g} = \mathbf{0}$ if their values and their Jacobian matrices are identical at that point. As shown in Propositions 6 and 7, the first condition, $\hat{\boldsymbol{\lambda}}^{\text{MD}}(\mathbf{0}; \eta) = \hat{\boldsymbol{\lambda}}^{\text{Bayes}}(\mathbf{0})$, holds for any $\eta$. The equivalence thus depends on matching their Jacobians.

From the proof of Proposition 6, the Jacobian of the MD estimator at $\boldsymbol{g} = \mathbf{0}$ is:

$$J_{\hat{\boldsymbol{\lambda}}^{\text{MD}}}(\mathbf{0}) = \frac{\eta}{m} \left( \boldsymbol{I} - \frac{1}{m} \mathbf{1}\mathbf{1}^T \right).$$

From the proof of Proposition 7, the Jacobian of the Bayesian posterior mean is:

$$J_{\hat{\boldsymbol{\lambda}}^{\text{Bayes}}}(\mathbf{0}) = \frac{1}{m(m+1)} \left( \boldsymbol{I} - \frac{1}{m} \mathbf{1}\mathbf{1}^T \right).$$

Equating the two Jacobians, $J_{\hat{\boldsymbol{\lambda}}^{\text{MD}}}(\mathbf{0}) = J_{\hat{\boldsymbol{\lambda}}^{\text{Bayes}}}(\mathbf{0})$, requires their scalar coefficients to be equal, since the matrix factor is non-zero for $m > 1$:

$$\frac{\eta}{m} = \frac{1}{m(m+1)}.$$

Solving for $\eta$ yields the unique solution $\eta = \frac{1}{m+1}$. □

This theorem provides a theoretical justification for the performance of the one-step MD estimator. It demonstrates that this simple, non-iterative update is not merely a heuristic but a principled, first-order approximation of the Bayesian posterior mean under a linearized likelihood.

**Remark (The Role of $\eta$ and the Small-Gradient Assumption).** The first-order equivalence established in Theorem 3 holds in the regime where $\|\boldsymbol{g}\| \to 0$, as this is where the higher-order terms, $O(\|\boldsymbol{g}\|^2)$, are negligible. For many models, the gradient's magnitude, $\|\boldsymbol{g}\|$, scales with the amount of data (e.g., sequence length $T$), which appears to invalidate the approximation when the data size is large.

However, the one-step MD estimator, $\hat{\boldsymbol{\lambda}}^{\text{MD}} = \mathrm{softmax}(\eta \boldsymbol{g})$, can remain a well-behaved estimator even for large $\|\boldsymbol{g}\|$ if $\eta$ is scaled appropriately. The term $\eta\boldsymbol{g}$ determines the softmax behavior. If we set the learning rate to be inversely proportional to the signal strength, for instance $\eta(T) = \Theta(1/T)$, the norm of the argument, $\|\eta\boldsymbol{g}\|$, can remain bounded. This scaling prevents the softmax output from saturating and allows the estimator to remain sensitive to the information in $\boldsymbol{g}$.

On the Bayesian side, under the linearized likelihood, the posterior mean $\hat{\boldsymbol{\lambda}}^{\text{Bayes}}(\boldsymbol{g})$ lacks a scaling parameter analogous to $\eta$. For large $\|\boldsymbol{g}\|$, the posterior density $\exp(\langle \boldsymbol{g}, \boldsymbol{\lambda} \rangle)$ concentrates sharply at the vertex of the simplex maximizing the inner product with $\boldsymbol{g}$. In this case, the first-order approximation from Proposition 7 breaks down as the neglected $O(\|\boldsymbol{g}\|^2)$ term becomes dominant.

The equivalence in Theorem 3 should therefore be interpreted as a local consistency result at $\boldsymbol{g} = \mathbf{0}$. It shows that for low-signal scenarios, the MD update is a principled approximation to the Bayesian one, and it provides a theoretically grounded value for $\eta$ in that regime. The empirical success of the one-step estimator for large $T$ suggests that while the functional forms of the two estimators diverge beyond the first order, a properly tuned MD estimator may still serve as an effective proxy for the Bayesian posterior mean, motivating analytical frameworks beyond local Taylor expansions.

# I Learning-rate scaling via a Lipschitz (smoothness) constant

In our main analysis, we established a first-order equivalence between the one-step MD estimator and the Bayesian posterior mean in the regime of "no evidence," where the log-likelihood gradient $\boldsymbol{g} = \mathbf{0}$. However, for a sequence of length $T$, the magnitude of the gradient, $\|\boldsymbol{g}\|$, typically scales with $T$. This raises the question of how the learning rate of the one-step MD estimator, $\hat{\boldsymbol{\lambda}}^{\text{MD}} = \mathrm{softmax}(\eta \boldsymbol{g})$ should be scaled with $T$ to maintain good performance.

In this section, we provide a theoretical justification for scaling the learning rate as $\eta = \Theta(1/T)$. We demonstrate that the negative log-likelihood function, viewed as a loss function over the simplex, has a gradient that is Lipschitz continuous with a constant $L$ that grows linearly with the sequence length $T$. Standard optimization theory suggests setting the learning rate inversely proportional to this Lipschitz constant, i.e., $\eta \propto 1/L$, to ensure stable updates. We use the same notation as the before:

$$c_{t,g} = \pi(y_{t-g}, y_t), \qquad t = m+1, \ldots, T, \qquad g = 1, \ldots, m,$$

and $m$ is the number of lags.

**Assumption.** We assume there exists a constant

$$c_{\min} > 0$$

such that for every $t \in \{m+1, \ldots, T\}$ and every $g \in \{1, \ldots, m\}$,

$$c_{t,g} = \pi(y_{t-g}, y_t) \geq c_{\min}. \tag{35}$$

Because each $c_{t,g}$ is a conditional probability, we also have the trivial upper bound

$$c_{t,g} \leq 1 \qquad \text{for all } t, g. \tag{36}$$

Under these assumptions the denominators that appear in derivatives are uniformly bounded away from zero, and global, uniform bounds on the Hessian are valid.

**Lemma 1** (Hessian decomposition)**.** *For $\ell(\boldsymbol{\lambda}) = \sum_{t=m+1}^{T} \log\big(\sum_{g=1}^{m} \lambda_g c_{t,g}\big)$ the Hessian satisfies*

$$\nabla^2 \ell(\boldsymbol{\lambda}) = -\sum_{t=m+1}^{T} \frac{\mathbf{c}_t \mathbf{c}_t^\top}{\big(\sum_{g=1}^{m} \lambda_g c_{t,g}\big)^2}, \qquad \mathbf{c}_t := (c_{t,1}, \ldots, c_{t,m})^\top.$$

*Hence the Hessian of the negative log-likelihood $f(\boldsymbol{\lambda}) = -\ell(\boldsymbol{\lambda})$ is*

$$\nabla^2 f(\boldsymbol{\lambda}) = \sum_{t=m+1}^{T} \frac{\mathbf{c}_t \mathbf{c}_t^\top}{\big(\sum_{g=1}^{m} \lambda_g c_{t,g}\big)^2}.$$

*Proof.* Direct differentiation of $\ell(\boldsymbol{\lambda})$ yields the displayed formulas. □

Write $s_t(\boldsymbol{\lambda}) := \sum_{g=1}^{m} \lambda_g c_{t,g}$ and consequently the Hessian of the loss is

$$\nabla^2 f(\boldsymbol{\lambda}) = \sum_{t=m+1}^{T} \frac{\mathbf{c}_t \mathbf{c}_t^\top}{s_t(\boldsymbol{\lambda})^2}. \tag{37}$$

Each summand in equation 37 is a rank-one positive semi-definite matrix.

### I.1 Uniform operator-norm bound on the Hessian

We now derive a uniform bound on the spectral (operator) norm of $\nabla^2 f(\boldsymbol{\lambda})$ that depends linearly on $T - m$.

**Lemma 2** (Operator-norm bound)**.** *Under the standing assumption equation 35 (and equation 36), for every $\boldsymbol{\lambda}$ in the full simplex $\Delta_{m-1} = \{\boldsymbol{\lambda} \geq 0,\ \sum_g \lambda_g = 1\}$ we have*

$$\left\|\nabla^2 f(\boldsymbol{\lambda})\right\|_{\mathrm{op}} \leq (T-m)\,\frac{m}{c_{\min}^2}. \tag{38}$$

*Proof.* From equation 37 and subadditivity of the operator norm,

$$\left\|\nabla^2 f(\boldsymbol{\lambda})\right\|_{\mathrm{op}} = \Big\| \sum_{t=m+1}^{T} \frac{\mathbf{c}_t \mathbf{c}_t^\top}{s_t(\boldsymbol{\lambda})^2} \Big\|_{\mathrm{op}} \leq \sum_{t=m+1}^{T} \frac{\left\|\mathbf{c}_t \mathbf{c}_t^\top\right\|_{\mathrm{op}}}{s_t(\boldsymbol{\lambda})^2}.$$

For a rank-one matrix $\mathbf{u}\mathbf{u}^\top$ the operator norm equals $\|\mathbf{u}\|_2^2$. Hence

$$\left\|\mathbf{c}_t \mathbf{c}_t^\top\right\|_{\mathrm{op}} = \|\mathbf{c}_t\|_2^2 = \sum_{g=1}^{m} c_{t,g}^2.$$

Using equation 36 we get the upper bound $\|\mathbf{c}_t\|_2^2 \leq m \cdot 1^2 = m$ for every $t$.

For the denominator, by equation 35 and since $\sum_{g=1}^m \lambda_g = 1$,

$$s_t(\boldsymbol{\lambda}) = \sum_{g=1}^{m} \lambda_g c_{t,g} \geq \sum_{g=1}^{m} \lambda_g c_{\min} = c_{\min}.$$

Therefore for every $t$,

$$\frac{\left\|\mathbf{c}_t \mathbf{c}_t^\top\right\|_{\mathrm{op}}}{s_t(\boldsymbol{\lambda})^2} \leq \frac{m}{c_{\min}^2}.$$

Summing over $t = m+1, \ldots, T$ yields

$$\left\|\nabla^2 f(\boldsymbol{\lambda})\right\|_{\mathrm{op}} \leq \sum_{t=m+1}^{T} \frac{m}{c_{\min}^2} = (T-m)\,\frac{m}{c_{\min}^2},$$

which is the bound equation 38. □

### I.2 Improved bound at the center of the simplex

Here we show that evaluating exactly at the uniform vector $\boldsymbol{\lambda}^*$ yields a strictly better constant: the spectral norm of the Hessian at $\boldsymbol{\lambda}^*$ is bounded by $(T-m)\,m^2$. This gives a less pessimistic Lipschitz constant and hence a looser restriction on the conservative step-size $\eta$.

**Proposition 8** (Operator-norm bound at the uniform vector)**.** *Let $\boldsymbol{\lambda}^* = (1/m, \ldots, 1/m)$. For the loss $f(\boldsymbol{\lambda}) = -\ell(\boldsymbol{\lambda})$ we have*

$$\left\|\nabla^2 f(\boldsymbol{\lambda}^*)\right\|_{\mathrm{op}} \leq (T-m)\,m^2. \tag{39}$$

*In particular, the gradient $\nabla f$ is Lipschitz at $\boldsymbol{\lambda}^*$ with constant $L^* \leq (T-m)m^2$, and the conservative step-size choice $\eta \leq 1/L^*$ yields $\eta = \Theta(1/T)$ for fixed $m$.*

*Proof.* Recall the Hessian decomposition (Eq. equation 37):

$$\nabla^2 f(\boldsymbol{\lambda}) = \sum_{t=m+1}^{T} \frac{\mathbf{c}_t \mathbf{c}_t^\top}{s_t(\boldsymbol{\lambda})^2}, \qquad \mathbf{c}_t = (c_{t,1}, \ldots, c_{t,m})^\top, \quad s_t(\boldsymbol{\lambda}) = \sum_{g=1}^{m} \lambda_g c_{t,g}.$$

Evaluate at the uniform vector $\boldsymbol{\lambda}^*$. Then

$$s_t(\boldsymbol{\lambda}^*) = \frac{1}{m} \sum_{g=1}^{m} c_{t,g} =: \frac{S_t}{m}, \qquad S_t := \sum_{g=1}^{m} c_{t,g}.$$

Hence the $t$-th summand becomes

$$\frac{\mathbf{c}_t\mathbf{c}_t^\top}{s_t(\boldsymbol{\lambda}^*)^2} = \frac{\mathbf{c}_t\mathbf{c}_t^\top}{(S_t/m)^2} = \frac{m^2}{S_t^2}\,\mathbf{c}_t\mathbf{c}_t^\top.$$

Taking operator norms and using subadditivity,

$$\left\|\nabla^2 f(\boldsymbol{\lambda}^*)\right\|_{\mathrm{op}} \le \sum_{t=m+1}^{T} \frac{m^2}{S_t^2}\left\|\mathbf{c}_t\mathbf{c}_t^\top\right\|_{\mathrm{op}}.$$

For each $t$, $\|\mathbf{c}_t\mathbf{c}_t^\top\|_{\mathrm{op}} = \|\mathbf{c}_t\|_2^2$. Observe the elementary inequality

$$\|\mathbf{c}_t\|_2^2 \le \Big(\sum_{g=1}^{m} c_{t,g}\Big)^2 = S_t^2,$$

which holds because $(\sum_i a_i)^2 = \sum_i a_i^2 + 2\sum_{i<j} a_i a_j \ge \sum_i a_i^2$ for nonnegative $a_i$. Using this inequality we obtain, for every $t$,

$$\frac{m^2}{S_t^2}\,\|\mathbf{c}_t\|_2^2 \le \frac{m^2}{S_t^2}\,S_t^2 = m^2.$$

Summing over $t = m+1, \dots, T$ yields

$$\left\|\nabla^2 f(\boldsymbol{\lambda}^*)\right\|_{\mathrm{op}} \le \sum_{t=m+1}^{T} m^2 = (T-m)\,m^2,$$

which proves equation 39. The remaining claims follow immediately from the standard equivalence between Hessian operator-norm bounds and local Lipschitz continuity of the gradient, and the reciprocal step-size rule $\eta \le 1/L^*$. □

**Theorem 4** (Lipschitz gradient and $\eta$ scaling at the uniform mixture)**.** *At the uniform vector $\boldsymbol{\lambda}^* = (1/m, \dots, 1/m)$ the loss $f(\boldsymbol{\lambda}) = -\ell(\boldsymbol{\lambda})$ is $L^*$–smooth with*

$$L^* \;\le\; (T-m)\,m^2. \tag{40}$$

*Consequently, the conservative step-size rule $\eta \le \frac{1}{L^*}$ yields the asymptotic scaling*

$$\eta \;=\; \Theta\Big(\frac{1}{T}\Big) \tag{41}$$

*for fixed $m$.*

*Proof.* Proposition 8 establishes that at $\boldsymbol{\lambda}^*$ the Hessian satisfies

$$\|\nabla^2 f(\boldsymbol{\lambda}^*)\|_{\mathrm{op}} \;\le\; (T-m)\,m^2.$$

By the standard equivalence between bounded Hessian operator norm and gradient Lipschitz continuity, this implies $\nabla f$ is $L^*$-Lipschitz at $\boldsymbol{\lambda}^*$ with $L^*$ as in equation 40. The conservative step-size choice $\eta \le 1/L^*$ is therefore sufficient to guarantee stability of gradient-based updates (and analogously for mirror descent / exponentiated-gradient after translating to the mirror geometry). Since $T - m = \Theta(T)$, the scaling $\eta = \Theta(1/T)$ follows for fixed $m$. □

The assumption equation 35 (strict positivity of every transition probability appearing in the likelihood) is the minimal condition that guarantees a *uniform* finite Lipschitz constant $L$ over the entire simplex $\Delta_{m-1}$. If some transitions were zero, then for parameter vectors placing mass on coordinates corresponding to zero transitions some denominators $s_t(\boldsymbol{\lambda})$ could vanish and the Hessian operator norm would be unbounded (hence no global $L$ exists).

# J Asymptotic Properties of the Likelihood Gradient

In this section, we study the asymptotic properties of the gradient of the log-likelihood function. We are mostly interested in the asymptotic scaling of the gradient with the sequence length $T$.

**Lemma 3** (Asymptotic Properties of the Gradient)**.** *Let the true parameter $\boldsymbol{\lambda}^* \in int(\Delta_{m-1})$ induce a Markov chain on the history space $\mathcal{Y}^m$ that is aperiodic and irreducible on a finite state space. This implies the chain is geometrically ergodic. Let $\mathbb{E}^{stat}_{\boldsymbol{\lambda}^*}$ denote expectation with respect to the stationary distribution of this chain. The mean of the score vector $\mathbf{g}(\mathbf{Y})$ exhibits the following asymptotic properties as $N_{obs} \to \infty$: The expected score vector scales linearly with $N_{obs} = T - m$:*

$$\mathbf{g}_0(\boldsymbol{\lambda}^*) \coloneqq \mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{g}(\mathbf{Y})] = N_{obs} \cdot \mathbf{v}(\boldsymbol{\lambda}^*) + O(1), \tag{42}$$

*where $\mathbf{v}(\boldsymbol{\lambda}^*)$ is the constant vector of stationary expected single-step scores, whose $h$-th component is:*

$$[\mathbf{v}(\boldsymbol{\lambda}^*)]_h = m \cdot \mathbb{E}^{stat}_{\boldsymbol{\lambda}^*}\left[\frac{\pi(Y_{t-h}, Y_t)}{\sum_{j=1}^{m} \pi(Y_{t-j}, Y_t)}\right]. \tag{43}$$

*The $O(1)$ term represents a constant offset due to initial conditions that does not grow with $N_{obs}$.*

*Proof.* Let us first define the score contribution from a single time step $t$ as the vector $\mathbf{z}_t(\mathbf{Y}) \in \mathbb{R}^m$, whose $h$-th component is given by:

$$[\mathbf{z}_t(\mathbf{Y})]_h = m\frac{\pi(Y_{t-h}, Y_t)}{\sum_{j=1}^{m} \pi(Y_{t-j}, Y_t)}.$$

The total score vector is the sum of these contributions over the observation period:

$$\mathbf{g}(\mathbf{Y}) = \sum_{t=m+1}^{n} \mathbf{z}_t(\mathbf{Y}), \quad \text{where } N_{\text{obs}} = T - m.$$

The process $(\mathbf{z}_t)_{t>m}$ is a sequence of random vectors. Because it is a function of the underlying ergodic Markov chain $(Y_{t-m}, \dots, Y_t)$, the sequence $(\mathbf{z}_t)$ is also ergodic and its distribution converges to a stationary distribution. By the linearity of expectation, the expected score is the sum of the individual expectations:

$$\mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{g}(\mathbf{Y})] = \sum_{t=m+1}^{n} \mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{z}_t(\mathbf{Y})].$$

The key assumption is the geometric ergodicity of the Markov chain. This implies that the distribution of the state $(Y_{t-m}, \dots, Y_t)$ converges exponentially fast to the unique stationary distribution, regardless of the initial state $(Y_1, \dots, Y_m)$. Consequently, the expectation $\mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{z}_t(\mathbf{Y})]$ converges exponentially fast to its stationary-state expectation, $\mathbf{v}(\boldsymbol{\lambda}^*)$. This convergence can be quantified. There exist a constant vector $\mathbf{C}$ and a rate $\rho \in (0, 1)$ such that for all $t > m$:

$$\|\mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{z}_t(\mathbf{Y})] - \mathbf{v}(\boldsymbol{\lambda}^*)\| \leq \|\mathbf{C}\|\rho^{t-m}.$$

We can now rewrite the sum of expectations:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{g}(\mathbf{Y})] &= \sum_{t=m+1}^{n} (\mathbf{v}(\boldsymbol{\lambda}^*) + (\mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{z}_t(\mathbf{Y})] - \mathbf{v}(\boldsymbol{\lambda}^*))) \\
&= \left(\sum_{t=m+1}^{n} \mathbf{v}(\boldsymbol{\lambda}^*)\right) + \left(\sum_{t=m+1}^{n} (\mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{z}_t(\mathbf{Y})] - \mathbf{v}(\boldsymbol{\lambda}^*))\right) \\
&= N_{\text{obs}} \cdot \mathbf{v}(\boldsymbol{\lambda}^*) + \mathbf{E}_n,
\end{aligned}$$

where $\mathbf{E}_n$ is the cumulative error term due to the process not having reached stationarity at early time steps. We can bound the norm of this error term:

$$\|\mathbf{E}_n\| = \left\| \sum_{t=m+1}^{n} (\mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{z}_t] - \mathbf{v}(\boldsymbol{\lambda}^*)) \right\| \leq \sum_{t=m+1}^{n} \|\mathbb{E}_{\boldsymbol{\lambda}^*}[\mathbf{z}_t] - \mathbf{v}(\boldsymbol{\lambda}^*)\| \leq \sum_{t=m+1}^{n} \|\mathbf{C}\|\rho^{t-m}.$$

Let $k = t - m$. The sum becomes a geometric series:

$$\|\mathbf{E}_n\| \leq \|\mathbf{C}\| \sum_{k=1}^{n-m} \rho^k < \|\mathbf{C}\| \sum_{k=1}^{\infty} \rho^k = \|\mathbf{C}\| \frac{\rho}{1-\rho}.$$

The sum of the error terms is bounded by a finite constant that does not depend on $n$ or $N_{\text{obs}}$. Therefore, the error term is $O(1)$. This proves the first claim: $\mathbf{g}_0(\boldsymbol{\lambda}^*) = N_{\text{obs}} \cdot \mathbf{v}(\boldsymbol{\lambda}^*) + O(1)$. Note that $O(1)$ is also $o(N_{\text{obs}})$, so the term is asymptotically sub-linear. □