Hyperparameter	**Value**
Base model	OLMo2-1B
Dataset	AlpacaEval (test split)
Max new tokens	1024
LM judge	Cohere Command A

Table 8: Hyperparameter configurations used for chat winrate on AlpacaEval.

B. Gradient Descent and Cross-entropy theory

B.1. Setup

Let s denote an individual instance (or input token sequence of language), with its true class identity (or the next token's index in vocabulary) given by $y(s)$. An (LLM) encoder, parameterized by θ, processes s to produce its contextualized embedding, $f_\theta(s) \in \mathbb{R}^d$, where d is the embedding dimension. For a batch, S, of b such instances, the encoder outputs a matrix $f_\theta(S) \in \mathbb{R}^{b\times d}$. Subsequently, predictions $\hat{\mathbf{y}} \in \mathbb{R}^{b\times|\mathcal{V}|}$ are generated by multiplying this batch embedding with a weight matrix $W \in \mathbb{R}^{d\times|\mathcal{V}|}$, where $|\mathcal{V}|$ is the vocabulary size (or number of classes):

$$\hat{\mathbf{y}} = f_\theta(S)W$$

To simplify the setting such that it is analytically-tractable, we assume the embedding function $f_\theta(s)$ to be modeled as a linear transformation of the input, i.e. $f_\theta(s) = \theta^T s$, where $\theta \in \mathbb{R}^{d_{in}\times d}$ is a parameter matrix. For a batch, of b such instances, represented as a matrix $S \in \mathbb{R}^{b\times d_{in}}$ whose row vectors are orthonormal (i.e., $SS^T = \mathbf{I}$). The batch embedding is, therefore, $f_\theta(S) = S\theta \in \mathbb{R}^{b\times d}$. Here, d_{in} is the input feature dimension and d is the embedding dimension.

Note: By imposing this i.i.d. assumption, we ensure that learning on one sample does not change the output of another sample, i.e. no inter-sample interference. While we admit that this assumption is unrealistic, and learning to predict the next token of one sequence in an autoregressive setup affects the output of another sequence, we believe that this assumption enables a first step towards understanding the implicit effect of cross-entropy loss optimization using gradient descent. We note that our results do not strictly depend on this assumption, and can be extended to non-i.i.d. samples. We leave this to future work.

Note 2: Note that our assumption of a linear embedding model, f_θ, is also an aberration from the transformer-based LLMs. However, we focus on the effect of loss and optimization in this section and leave further investigation into the implicit bias of architecture to future studies.

B.2. Linear approximation of Cross-entropy loss: Legendre Transform

Let us start by defining the cross-entropy loss for one example, s, which belongs to class c as:

$$\mathcal{L}_{CE}(s) = -\log\left(\frac{e^{\hat{y}_c}}{\sum_j e^{\hat{y}_j}}\right) = -\hat{y}_c + \log\left(\sum_j e^{\hat{y}_j}\right) \tag{6}$$

Note that eq. (6) is nonlinear in $\hat{\mathbf{y}}$, thereby making it harder to analyze the dynamical system in the parameter space that is imposed by gradient descent. In order to arrive at an analytical understanding of the gradient-induced dynamics when optimizing eq. (6), we will do a linear approximation of $\mathcal{L}_{CE}$ using Legendre Transform, similar to (Pezeshki et al., 2021). Specifically, we will derive the Legendre transform of the nonlinear term, $\log(\sum_j e^{\hat{y}_j})$.

Intuitively, we want to approximate eq. (6) such that it changes linearly with changes in $\hat{\mathbf{y}}$. The key motivation for using Legendre transform is to ignore the second (and higher) order effects of a "small" change in $\hat{\mathbf{y}}$ on $\mathcal{L}_{CE}$. Formally, we want the following:

$$\hat{\mathcal{L}}_{CE}(s) = -\hat{y}_c + \alpha^T\hat{\mathbf{y}} + g(\alpha) \tag{7}$$

where α is the slope of the nonlinear term at $\mathbf{x} = \hat{\mathbf{y}}$.

$$\alpha = \nabla_{\mathbf{x}} log(\sum_j e^{x_j})\Big|_{\mathbf{x}=\hat{\mathbf{y}}}$$

$$\implies \alpha_i = \frac{\partial}{\partial x_i} \log(\sum_j e^{x_j})\Big|_{x_j=\hat{y}_j} = \frac{e^{\hat{y}_i}}{\sum_j e^{\hat{y}_j}} \tag{8}$$

Note that: $\boxed{\sum_i \alpha_i = 1}$

To simplify things, let us denote $C = \sum_j e^{\hat{y}_j}$. Substituting this in eq. (8), $\hat{y}_i = log(\alpha_i) + log(C)$.

Now we need to find $g(\alpha)$ such that $f(\hat{\mathbf{y}}) = \alpha^T \hat{\mathbf{y}} + g(\alpha)$.

$$\begin{aligned} g(\alpha) = f(\hat{\mathbf{y}}) - \alpha^T \hat{\mathbf{y}} = \log(\sum_j e^{\hat{y}_j}) - \alpha^t \hat{\mathbf{y}} &= \log(\sum_j e^{\hat{y}_j}) - \sum_j \alpha_j \hat{y}_j \\ &= log(C) - \sum_j \alpha_j log(\alpha_j) - \sum_j \alpha_j log(C) \\ &= log(C) - \sum_j \alpha_j log(\alpha_j) - log(C) \qquad [\text{Using} \sum_j \alpha_j = 1] \\ &= -\sum_j \alpha_j log(\alpha_j) = H(\alpha) \end{aligned} \tag{9}$$

where $H(\alpha)$ denotes the Shannon entropy of a probability distribution defined by α_i's.

Substituting eq. (9) in eq. (7), we get the expression for the linearized cross-entropy loss:

$$\hat{\mathcal{L}}_{CE}(s) = -\hat{y}_c + \alpha^T \hat{\mathbf{y}} + H(\alpha) \tag{10}$$

B.3. Gradient descent dynamics of linearized cross-entropy loss

$$\begin{aligned} \hat{\mathcal{L}}_{CE}(s) = -\hat{y}_c + \alpha^T \hat{\mathbf{y}} + H(\alpha) &= -\hat{y}_c + \sum_j \alpha_j \hat{y}_j + H(\alpha) \\ &= -f_\theta(s)^T w_c + \sum_j \alpha_j f_\theta(s)^T w_j + H(\alpha) \\ \nabla_{f_\theta(s)} \hat{\mathcal{L}}_{CE}(s) &= -w_c + \sum_j \alpha_j w_j = \sum_j (\alpha_j - \delta_{j=c}) w_j \\ \nabla_{w_i} \hat{\mathcal{L}}_{CE}(s) &= -f_\theta(s)\delta_{i=c} + \alpha_i f_\theta(s) = (\alpha_i - \delta_{i=c}) f_\theta(s) \end{aligned}$$

where $\delta_{(.)}$ is the Dirac-delta function, i.e. its value is 1 when the condition in subscript is true and 0 otherwise.

Denoting $\tilde{\alpha}_i = (\alpha_i - \delta_{i=c})$, we arrive at the gradient equations for $f_\theta(s)$ and w_i's:

$$\nabla_{f_\theta(s)} \hat{\mathcal{L}}_{CE}(s) = \sum_j \tilde{\alpha}_j w_j \quad , \quad \nabla_{w_i} \hat{\mathcal{L}}_{CE}(s) = \tilde{\alpha}_i f_\theta(s) \tag{11}$$