DPO KERNELS - A Semantically-Aware, Kernel-Enhanced, and Divergence-Rich Paradigm for Direct Preference Optimization

Amitava Das[1*], Suranjana Trivedy[2], Danush Khanna[3], Rajarshi Roy[4], Gurpreet Singh[2†], Basab Ghosh[5], Yaswanth Narsupalli[6], Vinija Jain[7‡], Vasu Sharma[7‡], Aishwarya Naresh Reganti[8§], Aman Chadha[8§]

[1]BITS Pilani, Goa, [2]Artificial Intelligence Institute, University of South Carolina
[3]Manipal University Jaipur, [4]Kalyani Government Engineering College,
[5]IIITDM Kancheepuram, [6]IIT Kharagpur, [7]Meta AI, USA, [8]Amazon AI, USA

Abstract

The rapid advancement of large language models (LLMs) has revolutionized numerous applications, but presents significant challenges in aligning these models with diverse human values, ethical standards, and specific user preferences. Direct Preference Optimization (DPO) has become a cornerstone for preference alignment but is constrained by reliance on fixed divergence measures and limited feature transformations. We introduce **DPO-Kernels**, an innovative enhancement of DPO that integrates kernel methods to overcome these challenges through four key contributions: (i) **Kernelized Representations**: These representations enhance divergence measures by using polynomial, radial basis function (RBF), Mahalanobis, and spectral kernels for richer feature transformations. Additionally, we introduce a **hybrid loss** that combines embedding-based loss with probability-based loss; (ii) **Divergence Alternatives**: Beyond Kullback–Leibler (KL), we incorporate Jensen-Shannon, Hellinger, Rényi, Bhattacharyya, Wasserstein, and other f-divergences to boost stability and robustness; (iii) **Data-Driven Selection**: Choosing the optimal kernel-divergence pair among 28 combinations (4 kernels $\times$ 7 divergences) is challenging. We introduce automatic metrics that analyze the data to select the best kernel-divergence pair, eliminating the need for manual tuning; (iv) **Hierarchical Mixture of Kernels (HMK)**: Combining local and global kernels for precise and large-scale semantic modeling. This approach automatically selects the optimal kernel mixture during training, enhancing modeling flexibility. DPO-Kernels achieve state-of-the-art generalization in factuality, safety, reasoning, and instruction following across 12 datasets. While alignment risks overfitting, Heavy-Tailed Self-Regularization (HT-SR) theory confirms that DPO-Kernels ensure robust generalization in LLMs. Comprehensive resources are available to facilitate further research and application of DPO-Kernels.

1 DPO Revisited: Avenues for Advancement

The Direct Preference Optimization (DPO) (Rafailov et al., 2024) framework aims to optimize a policy $\pi(y \mid x)$ by balancing two objectives: improving the policy's ranking on preferred outcomes and regularizing it against a reference distribution using the Kullback–Leibler (KL) divergence. The DPO objective can be expressed as:

$$\max_{\pi} \underbrace{\mathbb{E}_{x,y^+,y^-}\left[\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right]}_{\text{Contrastive Loss}} - \underbrace{\alpha\, \mathbb{E}_x\left[\sum_y \pi(y \mid x) \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right]}_{\text{KL Divergence}}$$

where: x: The input prompt or context; y^+: The preferred output (e.g., a response chosen by human evaluators); y^-: The less preferred output, $\pi(y \mid x)$: The policy being optimized; $\pi_{\text{ref}}(y \mid x)$: The reference policy (often a pre-trained model's distribution); $\alpha > 0$: Hyperparameters controlling the strength of the regularization.

[*]Spearheaded work from conception to execution.
[†]Work done as part of research internship at AIISC.
[‡] Work done outside of role at Meta.
[§] Work done outside of role at Amazon.

DPO - Kernels (at-a-glance)

- **Representation**: We enrich the representation space by combining the standard probability-based contrastive loss with semantic embeddings, ensuring that model preferences reflect both statistical likelihoods and meaningful, context-sensitive qualities. (cf. Sec. 2) and Appendix D.
- **Kernels**: We enhance the DPO contrastive loss maximization by integrating kernel-based measures, allowing for flexible alignment in transformed feature spaces rather than relying solely on direct distribution comparisons. Incorporating polynomial, RBF, spectral, and Mahalanobis kernels. (cf. Sec. 3 and Appendix E).
- **Divergence**: Exploration of alternative divergence measures (e.g., Jensen-Shannon, Hellinger, Rényi, Bhattacharyya, Kullback-Leibler, Wasserstein, and f-divergences) addresses known limitations of KL divergence, such as instability and lack of robustness (cf. Sec. 4 and Appendix F).
- **Proposed DPO-Kernels**: DPO-kernels can be explained using a dual-objective framework that enhances the model's discriminative power while ensuring distributional stability:

$$\max_{\pi} \underbrace{\mathbb{E}_{x,y^+,y^-} \kappa \left[\overbrace{\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}}^{\text{Contrastive Loss}} + \overbrace{\gamma \log \left(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x} \right)}^{\text{Embedding Based Loss}} \right]}_{\text{Kernelized Hybrid Loss}} - \alpha\, \underbrace{\mathbb{E}_x \left[\sum_y \pi(y \mid x) \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \right]}_{\text{KL Divergence}}$$

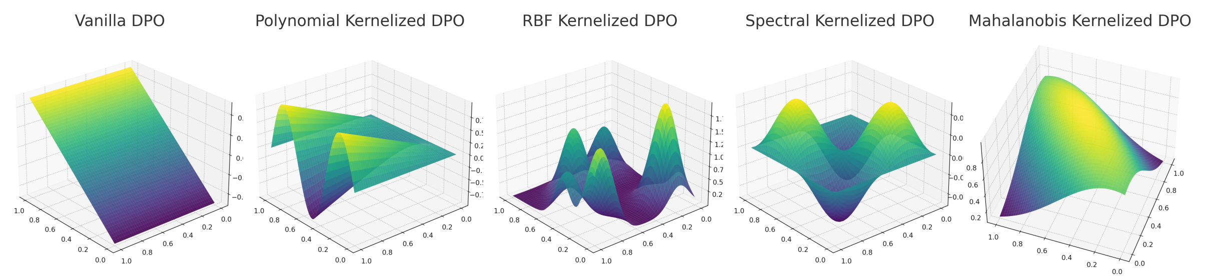
  The equation maximizes the Kernelized Contrastive Loss, which differentiates positive and negative samples using probability ratios and embedding similarities. Concurrently, it incorporates an Alternative Divergence Regularizer scaled by α, which enforces the model's distribution $\pi_\theta(y \mid x)$ to remain close to a reference distribution $\pi_{\text{ref}}(y \mid x)$ using a generic divergence measure D.
- **Data-Driven Selection of Kernel Type and Divergence Functions**: Selecting the best kernel-divergence pair from 28 combinations (4 kernels × 7 divergences) is non-trivial. To simplify this, we propose 4 metrics for kernel selection—*Positive-Negative Divergence (PND)*, *Positive-Negative Alignment Variance (PNAV)*, *Triplet Alignment Tightness (TAT)*, and *Normalized Alignment Gap (NAG)*—and 4 metrics for divergence selection: *Support Overlap*, *Drift Magnitude*, *Kurtosis*, and *Smoothness*. (cf. Sec. 5 and Appendix G).
- **Kernel Mixture and HMK Introduction:** The diversity of alignment tasks necessitates a kernel mixture model to leverage the complementary strengths of different kernels, such as local (e.g., RBF) and global (e.g., Spectral) patterns. However, naive mixtures are prone to kernel collapse, where one kernel dominates, reducing adaptability and generalization. To address this, we propose the **Hierarchical Mixture of Kernels (HMK)**, a robust framework that balances fine-grained and large-scale dependencies, maintaining kernel diversity and ensuring optimal alignment. (cf. Sec. 6 and Appendix H).
- **Empirical Findings:** Evaluations on 12 datasets show that **DPO-Kernels**, particularly HMK, achieve state-of-the-art generalization in factuality, safety, reasoning, and instruction-following tasks. However, HMK incurs 3-4× higher computational costs compared to standard DPO. We outline strategies to address this challenge in the limitations section, paving the way for cost-efficient future implementations. (cf. Sec. 7 and Appendix J).
- **Heavy-Tailed Self-Regularization (HT-SR):** Grounded in HT-SR theory, the *Weighted Alpha* metric (Martin et al., 2021a) provides a novel framework to evaluate generalization and overfitting in LLMs without relying on training or test data. Our analysis explores whether aligned models, particularly HMK, exhibit overfitting and quantifies the extent if present. (cf. Sec. 7.3 and Appendix N).
- **Broader Impact**: DPO-Kernels could transform AI alignment with human preferences, with future applications in text-to-image (Yoon et al., 2024; Wallace et al., 2023; Liu et al., 2024), text-to-video (Yoon et al., 2024), and Vision-Language Models (Wang et al., 2024; Yu et al., 2024).

Contrastive Loss $\left(\log \frac{\pi(y^+|x)}{\pi(y^-|x)}\right)$ encourages the policy π to assign higher probabilities to preferred outputs y^+ compared to less preferred outputs y^-, given the same input x. This term effectively pushes the policy to rank preferred responses higher, aligning it with observed preferences.

KL Divergence $\left(\sum_y \pi(y \mid x) \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\right)$ measures the divergence between the optimized policy π and the reference policy π_{ref}. This regularization term acts as a safeguard, preventing π from deviating excessively from the stable baseline provided by π_{ref}. Without this regularization, the policy might become overconfident in certain responses or drastically alter its distribution in undesirable ways. The hyperparameter α controls the strength of this regularization: a higher α keeps the policy closer to π_{ref}, making it more conservative, while a lower α allows greater flexibility for the policy to adjust probabilities based on preferences.

2 Richer Representation: Hybrid Approach: Integrating Probability and Embeddings

DPO (Rafailov et al., 2024) relies on the contrastive loss $\log \frac{\pi(y^+|x)}{\pi(y^-|x)}$, which focuses solely on probability-based preferences. While effective, this approach often neglects deeper semantic and qualitative factors inherent in human preferences. To address this limitation, we introduce a hybrid preference alignment method that integrates embedding-based signals alongside probability-based cues. Our approach defines a preference signal as $f_{\text{embed}}(x, y^+, y^-) = e_{y^+} - e_{y^-}$, where e_{y^+} and e_{y^-} are embedding-based similarity scores for positive and negative responses, respectively. For our experiments, we utilize `jina-embeddings-v3`



Kernel	Probability-Based and Embedding-Based Terms with Description
Polynomial	$\kappa\left[\log\left(\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)\right] = \left(\log\frac{\pi(y^+)}{\pi(y^-)} + c\right)^d$, $\kappa\left[\log\left(\frac{e_{y^+}\mid e_x}{e_{y^-}\mid e_x}\right)\right] = \left(\frac{(e_x^\top)e_{y^+}+c}{(e_x^\top)e_{y^-}+c}\right)^d$ Captures higher-order interactions using $(u^\top v + c)^d$. The parameter d controls complexity.
RBF	$\kappa\left[\log\left(\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)\right] = \exp\left(-\frac{\left(\log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)^2}{2\sigma^2}\right)$, $\kappa\left[\log\left(\frac{e_{y^+}\mid e_x}{e_{y^-}\mid e_x}\right)\right] = \exp\left(-\frac{\left(\frac{(e_x^\top)e_{y^+}}{(e_x^\top)e_{y^-}}\right)^2}{2\sigma^2}\right)$ Measures local similarity between inputs and outputs using the RBF kernel. σ controls smoothness.
Spectral	$\kappa\left[\log\left(\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)\right] = \sum_{i=1}^{p}\exp\left(-\lambda_i\left(\log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)^2\right)\phi_i\left(\log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)$, $\kappa\left[\log\left(\frac{e_{y^+}\mid e_x}{e_{y^-}\mid e_x}\right)\right] = \sum_{i=1}^{p}\exp\left(-\lambda_i\left(\frac{(e_x^\top)e_{y^+}}{(e_x^\top)e_{y^-}}\right)^2\right)\phi_i\left(\frac{(e_x^\top)e_{y^+}}{(e_x^\top)e_{y^-}}\right)$ Decomposes inputs and outputs into eigenfunctions ϕ_k and eigenvalues λ_k to capture global, frequency-based dependencies.
Mahalanobis	$\kappa\left[\log\left(\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)\right] = \exp\left(-\frac{\left(\log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}-\mu\right)^2}{2\sigma^2}\right)$, $\kappa\left[\log\left(\frac{e_{y^+}\mid e_x}{e_{y^-}\mid e_x}\right)\right] = \exp\left(-\frac{\left(\frac{(e_x^\top)e_{y^+}}{(e_x^\top)e_{y^-}}-\mu'\right)^2}{2\sigma'^2}\right)$ Leverages the Mahalanobis distance to capture anisotropic feature correlations using the covariance matrix Σ.
HMK	$\kappa\left[\log\left(\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)\right] = \sum_{i=1}^{4}\tau_i\lambda_i\kappa_i\left(\log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\right)$, $\kappa\left[\log\left(\frac{e_{y^+}\mid e_x}{e_{y^-}\mid e_x}\right)\right] = \tau_1\left(\frac{\lambda_1\kappa_{\text{RBF}}(e_x,e_{y^+})+\lambda_2\kappa_{\text{Poly}}(e_x,e_{y^+})}{\lambda_1\kappa_{\text{RBF}}(e_x,e_{y^-})+\lambda_2\kappa_{\text{Poly}}(e_x,e_{y^-})}\right) + \tau_2\left(\frac{\lambda_3\kappa_{\text{Spectral}}(e_x,e_{y^+})+\lambda_4\kappa_{\text{Maha}}(e_x,e_{y^+})}{\lambda_3\kappa_{\text{Spectral}}(e_x,e_{y^-})+\lambda_4\kappa_{\text{Maha}}(e_x,e_{y^-})}\right)$ Combines multiple kernels hierarchically, balancing local kernels (RBF, Polynomial) and global kernels (Spectral, Mahalanobis). $K(x,x') = \tau_1(\lambda_1 K_{\text{RBF}} + \lambda_2 K_{\text{Poly}}) + \tau_2(\lambda_3 K_{\text{Spectral}} + \lambda_4 K_{\text{Maha}})$

Table 1: Kernel methods implicitly map data into a high-dimensional feature space using kernels that compute inner products in this transformed space. For more details on gradient descent dynamics in kernel-induced loss landscapes, see Appendix K. This table summarizes the kernelized hybrid loss into (a) kernelized probability-based loss and (b) kernelized embedding-based loss for Polynomial, RBF, Spectral, Mahalanobis kernels, and HMK.

(Sturua et al., 2024), but the framework is adaptable to other embeddings, enabling generalization across embedding models. To compute the embedding-based similarity scores, we apply cosine similarity: $\text{sim}(e_x, e_y) = \frac{e_x^\top e_y}{\|e_x\|\|e_y\|}$, where e_x is the embedding for the prompt and e_y is for the response. These similarity scores are normalized using a softmax temperature parameter τ, such that: $> \text{sim}^*(e_x, e_y) = \frac{\exp(\text{sim}(e_x,e_y)/\tau)}{\sum_{y'}\exp(\text{sim}(e_x,e_{y'})/\tau)}$. In practice, we set $\tau = 0.07$ to accentuate semantic differences in embedding space.

Embedding-based representations are well-established in preference modeling, reward design, and metric learning (Bai et al., 2022b; Ouyang et al., 2022; Peyré and Cuturi, 2019), often relying on pairwise distances or fixed objectives (Oord et al., 2018; Chen et al., 2020; Radford et al., 2021). Recent large language models (LLMs) like LaMDA (Thoppilan et al., 2022) and PaLM (Chowdhery et al., 2022) also leverage embeddings for preference alignment. However, existing approaches typically treat embeddings and probability-based signals separately, relying on fixed divergence measures (e.g., KL, triplet loss

(Schroff et al., 2015), or contrastive loss (Hadsell et al., 2006)). In contrast, our work is the **first to bridge embeddings and probability-based alignment in a unified parametric framework for policy learning**, offering a more comprehensive approach to preference optimization.

Hybrid Loss: We blend probability and embedding signals:

$$\max_\pi \underbrace{\mathbb{E}_{x,y^+,y^-}[\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + \gamma(\log \frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)})]}_{\text{Hybrid Loss}} - \alpha KL$$

with $\gamma > 0$ controlling the contribution of the embedding signal. When $\gamma = 0$, we recover the standard DPO loss. Increasing γ guiding the policy to produce outputs that are both probable and semantically preferable.

Interpretation:

- **Embedding-Guided Tie-Breaking**: When probabilities are similar, embeddings help break ties by favoring outputs that are semantically more aligned or orthogonal. This alignment ensures that the selected output is not only probable but also semantically relevant, which is crucial for preference-driven alignment.
- **Semantic Consistency Check**: If the model strongly prefers y^+ but embeddings do not support its semantic quality, a moderate γ prevents purely probability-driven reinforcement. Instead, it encourages the model to refine its output distribution to better align with semantic criteria, promoting more meaningful preference-based selection.

The hybrid loss is then embedded within a kernel function, enabling DPO-Kernel to capture local, global, and higher-order dependencies, as detailed in the next section. Appendix D formulates our novel hybrid loss covering its mathematical definition, term-based decomposition, properties, impact on policy learning, etc.

3 Kernel-Integrated DPO Formulation

Standard DPO aligns a policy π with human preferences while regularizing against a reference distribution π_{ref} via a divergence $D(\cdot\|\cdot)$. While effective, this approach relies on simple distributional differences, which may fail to capture deeper semantic relationships essential for alignment. To address this, we introduce kernelized proximity measures that enable more expressive and adaptive alignment. Our framework extends DPO into four distinct DPO-Kernel variants: (i) Polynomial, (ii) RBF, (iii) Spectral, and (iv) Mahalanobis. The resulting objective is expressed as:

$$\max_\pi \underbrace{\mathbb{E}_{x,y^+,y^-}\kappa\Big[\log\Big(\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\Big) + \gamma \log\Big(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x}\Big)\Big]}_{\text{Kernelized Hybrid Loss}} - \alpha KL$$

Each kernel offers a unique perspective on alignment. Polynomial kernels capture higher-order interactions, enabling compositional reasoning. RBF kernels emphasize local, fine-grained structure, useful for proximity-based alignment. Spectral kernels capture global, oscillatory patterns to handle periodic dependencies, while Mahalanobis kernels leverage feature covariance to account for anisotropic relationships. These kernelized variants preserve the core mathematical foundations of DPO while significantly enhancing its ability to capture richer alignment criteria. For each kernel type κ, we compute the kernelized loss by applying it independently to the contrastive and embedding components. Let $z = \log \frac{\pi(y^+|x)}{\pi(y^-|x)}$ and $z' = \log \frac{\text{sim}^*(e_x, e_{y^+})}{\text{sim}^*(e_x, e_{y^-})}$, then the combined loss becomes $\mathcal{L}(x, y^+, y^-) = \kappa(z) + \gamma\kappa(z')$. Gradients are computed using automatic differentiation in PyTorch. Each kernel is parameterized with differentiable components, e.g., degree d for Polynomial, bandwidth σ for RBF, and eigenvalue filters λ_i for Spectral.

Sec. 1 illustrates the effect of kernelizing the DPO objective with various kernels, including Polynomial, RBF, Spectral, and Mahalanobis, in comparison to the Vanilla DPO. Each plot shows how different kernels reshape the optimization landscape by implicitly mapping input data to higher-dimensional feature spaces, allowing the model to capture complex patterns and interactions. This kernelized transformation enhances the expressiveness of the DPO objective, enabling it to adapt to diverse data distributions and modeling needs.

4 Replacing KL regularizer with alternatives

The original DPO framework typically utilizes the Kullback–Leibler (KL) divergence to align the learned policy $\pi(y \mid x)$ with the reference distribution $p_{\text{ref}}(y \mid x)$. While KL divergence is favored for its strong theoretical foundations, exploring alternative divergence measures can lead to more robust optimization, enhanced stability, and improved interpretability and generalizability.

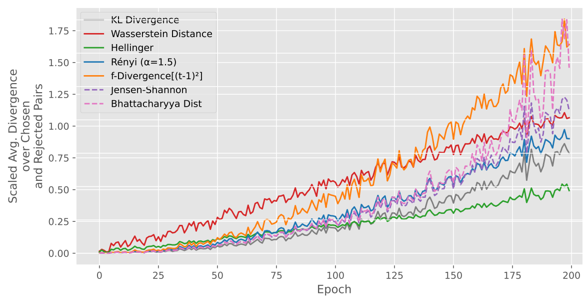


Figure 1: The plot illustrates the oscillatory behavior and trends of divergence measures, including Wasserstein, Jensen-Shannon, Hellinger, Rényi, KL, Bhattacharyya, and f-divergence, as training progresses, reflecting their sensitivity to the alignment dynamics.

Fig. 1 illustrates the temporal evolution of various divergence measures, including KL Divergence, Wasserstein Distance, Hellinger, Rényi, Bhattacharyya, Jensen-Shannon, and f-divergence, across training steps. The oscillatory behavior observed in the higher divergence measures (e.g., Rényi, Bhattacharyya, and f-divergence) highlights their sensitivity to dynamic alignment changes. In contrast, smoother trends in Wasserstein and Jensen-Shannon divergences indicate their stability and robustness over time. The overall upward trajectory reflects increasing distributional alignment shifts as training progresses, providing insights into how divergence measures respond to evolving alignment dynamics.

Each divergence function $D(\pi \| \pi_{\text{ref}})$ is computed over a mini-batch using Monte Carlo approximation: $D \approx \frac{1}{N}\sum_{i=1}^{N} d(\pi(y_i|x_i), \pi_{\text{ref}}(y_i|x_i))$, where $d(\cdot,\cdot)$ denotes the pointwise divergence. For Wasserstein distance, we use the Sinkhorn approximation with entropic regularization $\epsilon = 0.01$. The cost matrix for token-level Wasserstein is constructed from cosine embedding distances between token vectors.

The mathematical formulations and descriptions for each of the divergence functions are summarized in Table 5 in Appendix F.

5 Data-Driven Selection of Kernel Types and Divergence Functions

Choosing the optimal kernel-divergence pair among 28 combinations (4 kernels $\times$ 7 divergences) is challenging. We propose a systematic, data-driven framework that replaces heuristics with well-defined metrics, ensuring adaptability and improved generalization.

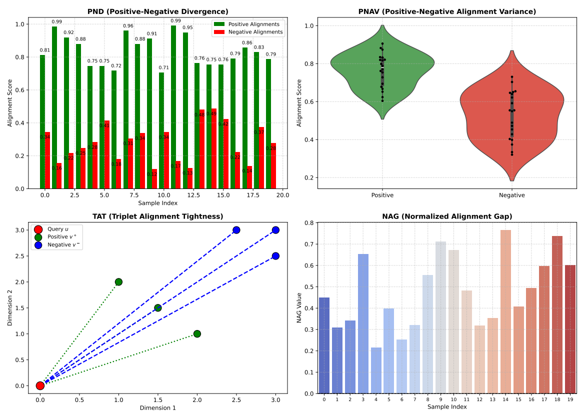


Figure 2: Visualization of the four proposed metrics for kernel selection in alignment tasks. **(a) PND** illustrates the divergence between alignment scores for positive and negative samples, indicating the degree of separability. **(b) PNAV** depicts the variance in alignment scores for positive and negative samples, reflecting alignment consistency. **(c) TAT** shows the relative positioning of query (x), positive (y^{+}), and negative (y^{-}) embeddings in the latent space, highlighting alignment precision. **(d) NAG** tracks the evolution of alignment gaps over samples, where smaller NAG values signify better alignment quality.

5.1 Data-Driven Kernel Selection Logic

We propose four novel metrics—*Positive-Negative Divergence (PND)*, *Positive-Negative Alignment Variance (PNAV)*, *Triplet Alignment Tightness (TAT)*, and *Normalized Alignment Gap (NAG)*—that quantify key geometric and relational properties of the data, summarized in Table 6 in Appendix G. Fig. 2 visualizes the four proposed metrics for kernel selection in alignment tasks: these metrics collectively assess alignment properties, such as separability, consistency, precision, and gap quality, enabling a comprehensive evaluation of kernel performance in alignment.

Here, we prescribe a practical guideline to help users empirically select the most suitable kernel for alignment tasks based on key metrics. By leveraging thresholds for metrics such as PNAV, TAT, NAG, and PND, this framework provides an intuitive yet effective approach to kernel selection, ensuring alignment properties are well-captured for diverse scenarios.

$$k^* = \begin{cases} \text{RBF Kernel}, & \text{if PNAV} > \varepsilon_1 \text{ and TAT} < \varepsilon_2 \\ \text{Polynomial Kernel}, & \text{if NAG} \approx 0 \text{ and PND} \approx 0 \\ \text{Mahalanobis Kernel}, & \text{if NAG} > 0 \text{ and PNAV} < \varepsilon_3 \\ \text{Spectral Kernel}, & \text{if TAT} > \varepsilon_4 \text{ and PND} < \varepsilon_5 \end{cases}$$

Here, thresholds $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5$ are empirically tuned or determined through validation. Initial values such as $\varepsilon_1 = 0.5$, $\varepsilon_2 = 0.3$, $\varepsilon_3 = 0.2$, $\varepsilon_4 = 0.7$, and $\varepsilon_5 = 0.1$ serve as practical defaults. Balanced metrics (e.g., ≈ 0) signal alignment structures, while larger deviations reveal more intricate relationships requiring advanced kernels.

To reduce computational load during metric evaluation, we sample 500 triplets per epoch and cache embedding distances. PNAV and TAT metrics are computed using vectorized operations in NumPy. We update kernel choice every 10 epochs to avoid oscillatory switching. In distributed settings, metrics are aggregated across nodes using AllReduce.

5.2 Data-Driven Divergence Choice Logic

We further propose four distributional metrics—*Support Overlap*, *Drift Magnitude*, *Kurtosis*, and *Smoothness*—to systematically select the most appropriate divergence measure, summarized in Table 7 in Appendix G. Furthermore, Fig. 3 in Appendix G visualizes the four proposed metrics for divergence selection: these metrics provide insights into the behavior of distributions by quantifying their overlap, shift, tail properties, and functional smoothness. Collectively, they enable the empirical selection of the most appropriate divergence measure for various data scenarios, ensuring effective modeling and comparison of distributions.

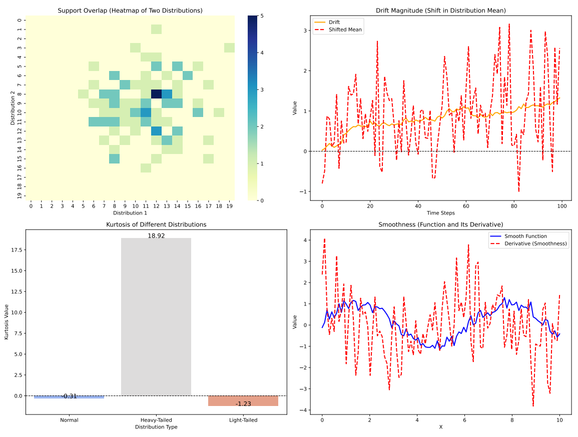


Figure 3: Visualization of the four key metrics for divergence selection: **(1) Support Overlap**: Heatmap representing the overlap between two distributions, highlighting shared support regions; **(2) Drift Magnitude**: Illustration of the shift in the mean of a distribution over time, showcasing how drift is detected; **(3) Kurtosis**: Bar plot comparing kurtosis values for normal, heavy-tailed, and light-tailed distributions, quantifying the "tailedness" of each distribution; **(4) Smoothness**: Visualization of a smooth function and its derivative, where smoother functions exhibit smaller, less abrupt changes in derivatives.

To compute Drift Magnitude, we track the moving average of the embedding mean over batches: $\Delta\mu = \|\mu_t - \mu_{t-1}\|_2$, where $\mu_t = \frac{1}{N}\sum_{i=1}^{N} e_{x_i}$. Kurtosis is estimated via: $\text{Kurt}(X) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - \bar{x}}{\sigma}\right)^4$, and smoothed using exponential moving average with decay rate $\beta = 0.9$. The Support Overlap heatmap is generated by thresholding token probabilities at $\epsilon = 10^{-4}$ and computing the Jaccard index over top-k sets.

We provide a practical guideline to help users empirically select the most suitable divergence measure based on key metrics. These metrics offer insights into distributional behavior, ensuring the chosen divergence measure aligns with the data's characteristics.

$$D^* = \begin{cases} \text{Bhattacharyya Divergence}, & \text{if Support Overlap} > \varepsilon_1 \\ \text{Wasserstein Divergence}, & \text{if Drift Magnitude} > \varepsilon_2 \\ \text{Rényi Divergence}, & \text{if Kurtosis} > \varepsilon_3 \\ \text{Jensen-Shannon Divergence}, & \text{if Overlap is low and Kurtosis is low} \\ \text{Hellinger Divergence}, & \text{if Smoothness is low and Kurtosis is low} \\ \text{KL Divergence}, & \text{otherwise} \end{cases}$$

We recommend starting with thresholds $\varepsilon_1 = 0.6$, $\varepsilon_2 = 0.3$, and $\varepsilon_3 = 3$, refining them based on the observed performance. This systematic approach ensures that divergence selection is directly tailored to the alignment complexity of the data. Appendix G offers a detailed discourse for data-driven selection of kernel types and divergence functions based on the appropriate metrics.

6 Kernel Mixture Approach - HMK

The use of a single kernel may fail to capture the diverse relationships in alignment tasks. Different kernels excel at modeling local similarities, global structures, or higher-order interactions, making it hard for any one to perform universally. A **Kernel Mixture Approach** overcomes this limitation by dynamically combining multiple kernels, leveraging their complementary strengths to improve generalization across varied datasets (e.g., diverse alignment tasks (Dubois et al., 2024b; Lv et al., 2023a), policy shifts (Koh et al., 2021a), and evolving alignment requirements (Jain et al., 2024a)).

Related Works: Research in multiple kernel learning (Gönen and Alpaydın, 2011), Gaussian processes (Duvenaud et al., 2013), and distributional adaptation (Quinonero-Candela et al., 2009; Koh et al., 2021b) shows that combining kernels effectively addresses dataset heterogeneity and distributional shifts. Inspired by these insights, we adopt a Kernel Mixture Approach expressed as: $\kappa(u, v) = \lambda_1 \kappa_{\text{poly}}(u, v) + \lambda_2 \kappa_{\text{RBF}}(u, v) + \lambda_3 \kappa_{\text{spec}}(u, v) + \lambda_4 \kappa_{\text{Maha}}(u, v)$, where $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$ and $\sum_{i=1}^{4} \lambda_i = 1$. The weights are set via softmax: $\lambda_i = \frac{\exp(\theta_i)}{\sum_{j=1}^{4} \exp(\theta_j)}$, with trainable parameters θ_i optimized through gradient descent. This formulation dynamically adapts kernel contributions to the task at hand.

However, a key challenge of this approach is **kernel collapse** (Lanckriet et al., 2004, 2002; Rätsch and Warmuth, 2005), where one kernel disproportionately dominates, effectively reducing the model to a single-kernel learner. This diminishes diversity and undermines the representational power needed to model complex data relationships.

To mitigate kernel collapse, we add an entropy-based regularizer to the kernel weights: $\mathcal{R}_{\text{entropy}} = -\sum_{i=1}^{4} \lambda_i \log \lambda_i$, encouraging weight diversity. This term is scaled by a coefficient $\beta = 0.1$. During training, we alternate between optimizing π and the kernel weights λ, using Adam optimizers with learning rates $1e{-}5$ and $5e{-}4$, respectively.

We analyze kernel-specific behavior and hyperparameter sensitivity in Appendix G,Appendix H, and Appendix N, including alignment trade-offs and implications for robustness. Addressing this issue is essential for fully realizing the potential of kernel mixtures in alignment tasks. For detailed discussion please refer to Appendix H.

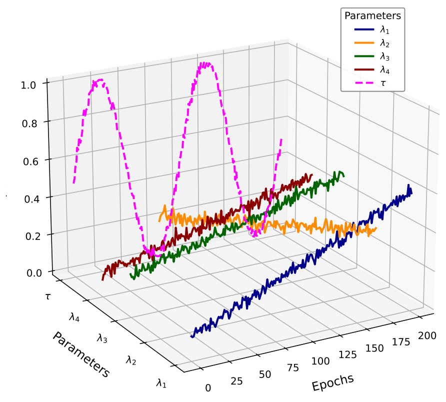


Figure 5: Dynamic evolution of kernel weights $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ and Local-Global Balance Coefficients (τ_1, τ_2). The model shifts its reliance on local or global kernels over training epochs, achieving a stable balance.

6.1 Hierarchical Mixture of Kernels (HMK)

Hierarchical Mixture of Kernels (HMK) overcomes kernel collapse by introducing a two-level

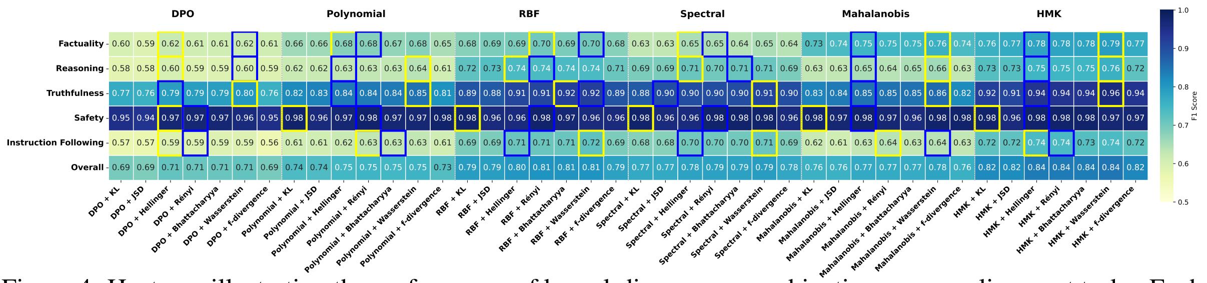


Figure 4: Heatmap illustrating the performance of kernel-divergence combinations across alignment tasks. Each row represents a task (Factuality, Reasoning, Truthfulness, Safety, Instruction Following), while the 'Overall' row aggregates average performance. For a task-specific breakdown (e.g., for RBF), see Appendix J and Fig. 19

decomposition that balances **local kernels** (RBF, Polynomial) (Schölkopf and Smola, 2002) and **global kernels** (Spectral, Mahalanobis) (Weinberger and Saul, 2009; Ng et al., 2001). Local kernels capture short-range dependencies, while global kernels model broader, long-range relationships. HMK assigns learnable weights to both groups, enabling dynamic adaptation to varying data geometries:

$$K(x,x') = \tau_1(\lambda_1 K_{\text{RBF}} + \lambda_2 K_{\text{Poly}}) + \tau_2(\lambda_3 K_{\text{Spectral}} + \lambda_4 K_{\text{Maha}}),$$

where τ_1, τ_2 balance local-global contributions. Both τ and λ are updated through backpropagation, allowing HMK to maintain kernel diversity and adapt effectively.

The hierarchical coefficients τ_1 and τ_2 are initialized to 0.5 and constrained via softmax to ensure $\tau_1 + \tau_2 = 1$. We use gradient clipping at 1.0 to stabilize training during the early epochs where and rapidly change. HMK is implemented as a custom PyTorch module supporting backpropagation through both kernel layers and balance parameters. Kernel evaluations are batched and fused using matrix operations to maximize GPU efficiency.

Fig. 5 shows the evolution of kernel weights $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ and Local-Global Balance Coefficients (τ_1, τ_2) over training. Early epochs highlight competition between local and global kernels, with τ_1 and τ_2 stabilizing around epoch 100. Polynomial (λ_1) and RBF (λ_2) dominate initially, while Spectral (λ_3) and Mahalanobis (λ_4) gain influence later, emphasizing global dependencies. By epoch 200, the system converges to an optimal balance. For detailed discussion please refer to Appendix H.

7 Empirical Results

We conducted all our experiments using Llama 3.3 (raymondd, 2024). Appendix C details our experiments and evaluation setup.

7.1 Datasets and Tasks

We assess the performance of models trained with DPO-Kernels across 12 diverse preference datasets, thoughtfully chosen to encompass a wide spectrum of data sources. These datasets are categorized as follows: I. **Human-Annotated Datasets**: HH-RLHF (Bai et al., 2022a), HelpSteer (Wang et al., 2023), Chatbot Arena 2023 (Zheng et al., 2023), Chatbot Arena 2024 (Chiang et al., 2024), AlpacaFarm Human (Dubois et al., 2024c), and PRM800k (Lightman et al., 2023). II. **Web-Scraped Datasets**: SHP-2 (Ethayarajh et al., 2022). III. **Synthetically Generated Datasets:** Ultra-Feedback (Cui et al., 2024), Nectar (Zhu et al., 2023), Orca (Lv et al., 2023b), Capybara (Daniele and Suphavadeeprasit, 2023a), and AlpacaFarm GPT-4 (Daniele and Suphavadeeprasit, 2023b). Collectively, these datasets span a broad range of alignment tasks, including Factuality, Reasoning, Truthfulness, Safety, and Instruction Following, thereby providing a comprehensive evaluation framework for the DPO-Kernels approach. Appendix B highlights the details of datasets used in this work, including human-annotated and syntehtically generated datasets.

7.2 Efficacy of Hybrid Loss and Divergence-Based Regularizers

Fig. 4 demonstrates how hybrid loss and divergence-based regularizers improve alignment across *Factuality, Reasoning, Truthfulness, Safety,* and *Instruction Following*. Compared to standard DPO, hybrid loss consistently achieves higher performance, with both *RBF* and *HMK* showing better performance. Moreover, *RBF* emerges as the best-performing single kernel overall, while *Rényi* and *Bhattacharyya* divergences excel in *Truthfulness* and *Instruction Following*. Notably, *Safety* performance remains robust across divergences (Fig. 19).

During training, we apply early stopping if validation alignment score does not improve for 5 consecutive checkpoints. All experiments use mixed-precision training (FP16) via PyTorch's `torch.cuda.amp` to accelerate training and reduce memory usage. We log divergence values and alignment metrics using Weights & Biases for visualization and reproducibility. Checkpoint averaging over the last 3 epochs is used for final evaluation to mitigate variance in model predictions.

Detailed ablations, including per-kernel performance, divergence combinations, and comparisons across 12 datasets, are provided in Appendix C and Appendix J.

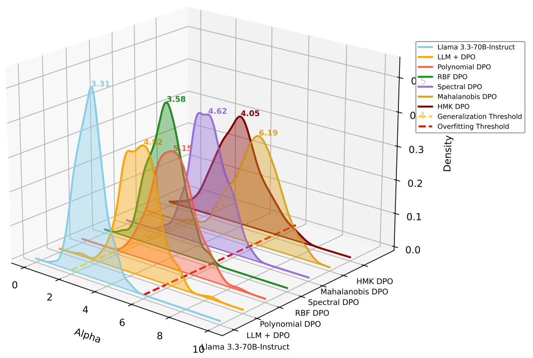


Figure 6: Generalization vs. overfitting trade-off for various DPO-kernels, grounded in HTSR theory. Smaller α values indicate stronger self-regularization and better generalization, while larger α values signal overfitting or under-optimized layers.

7.3 Generalization vs. Overfitting: Which Kernel/Divergence Excels?

1. RQ1: Do aligned LLMs lose generalizability and become overfitted? Alignment procedures slightly increase overfitting, with a generalization error drift $|\Delta\mathcal{E}_{\text{gen}}| \leq 0.1$ (within $\pm 10\%$), which is considered acceptable. Results reported in Fig. 6.

2. RQ2: Which kernel and divergence functions offer the best generalizability? RBF and Spectral kernels exhibit the least generalization, while Polynomial kernels increase overfitting by 15%. Mahalanobis kernels perform similarly to RBF and Spectral but incur higher computational costs. Among divergences, Bhattacharyya and Wasserstein yield the strongest generalization, outperforming KL and Jensen-Shannon. Rényi divergence is effective for specific tasks but requires careful tuning of α to balance alignment strength and overfitting risks. Appendix N details the theory behind Heavy-Tailed Self-Regularization (HT-SR) and the *Weighted Alpha* metric (Martin et al., 2021a), offering a statistical mechanics to measure generalization in DNN.

8 Conclusion

We introduced **DPO-Kernels**, a novel framework for alignment that combines **kernelized representations** with **divergence-based regularization**. Leveraging a *Hierarchical Mixture of Kernels (HMK)* and **data-driven selection**, our method addresses robust generalization and scalable alignment. Selecting the optimal kernel-divergence pair from **28 combinations** ($4\ k \times 7\ d$) is challenging, so we proposed data driven metrics. Evaluated on **12 datasets**, DPO-Kernels achieves *SoTA* performance across tasks. Although *HMK* outperforms, it incurs **3x-4x higher** computational costs than baseline DPO methods. Future work may use **Random Fourier Features (RFF)** or **Nyström methods** to reduce complexity.

9 Discussion and Limitations

While DPO-Kernels demonstrate significant advancements in alignment and generalization, several limitations warrant further attention.

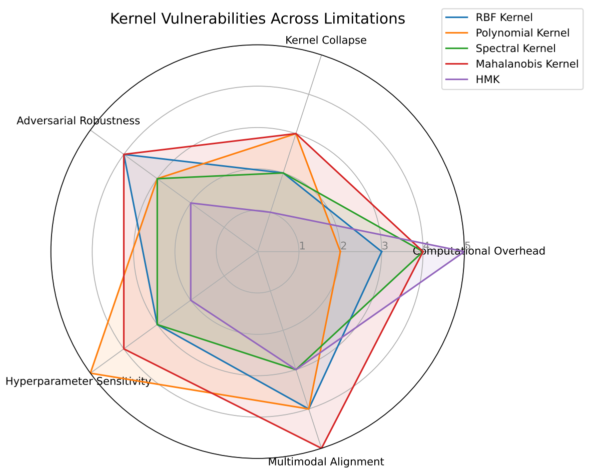


Figure 7: Radar chart illustrating the vulnerabilities of different kernels (RBF, Polynomial, Spectral, Mahalanobis) and the HMK framework across key limitations: *Computational Overhead*, *Kernel Collapse*, *Adversarial Robustness*, *Hyperparameter Sensitivity*, and *Multimodal Alignment*. Each axis represents a limitation, and the plotted values indicate the vulnerability severity on a scale of 1 (low vulnerability) to 5 (high vulnerability).

1. Computational Overhead: The Hierarchical Mixture of Kernels (HMK) incurs a computational cost 3-4x higher than baseline methods, primarily due to dynamic kernel balancing and hierarchical decomposition. Approximation techniques like Random Fourier Features (RFF) (Rahimi and Recht, 2007), Nyström methods (Williams and Seeger, 2001), and sparse Gaussian processes (Snelson and Ghahramani, 2006) can alleviate this overhead, making the framework more scalable for large-scale datasets. HMK's computational cost is justified by superior alignment capabilities.

2. Kernel Collapse: The dominance of a single kernel during training, known as kernel collapse, limits the diversity of kernel contributions. Mitigations include entropy-based regularization (Nemirovski et al., 2009) to promote kernel diversity and certified robustness (Wong and Kolter, 2018) to enforce balanced kernel contributions.

3. Adversarial Robustness: HMK's sensitivity to adversarial preference perturbations is currently untested. Small input changes can result in significant alignment shifts. Approaches such as adversarial training (Madry et al., 2018) and robust kernel learning (Xu et al., 2009) could strengthen resilience.

4. Hyperparameter Sensitivity: Performance depends on sensitive parameters like the RBF bandwidth (σ), Polynomial degree (d), and Mahalanobis covariance (Σ). Techniques such as meta-learning (Finn et al., 2017a) and adaptive tuning (Hazan et al., 2007) can streamline hyperparameter optimization.

5. Multimodal Alignment: Extending HMK to multimodal tasks (e.g., text-image alignment) involves computationally expensive cross-modal kernel computations. Techniques like cross-modal contrastive learning (Radford et al., 2021) and cross-modal RFF approximations could improve efficiency.

Addressing these limitations through the suggested mitigations will not only enhance the scalability and robustness of DPO-Kernels but also broaden their applicability to dynamic, multimodal alignment tasks. Refer to Table 2 and Fig. 7 for a detailed overview of limitations and solutions.

10 Ethical Considerations

The DPO-Kernels framework offers significant potential for alignment tasks, yet its application demands careful attention to ethical concerns. Below, we highlight key considerations and propose actionable strategies to address them.

10.1 Fairness and Bias

Kernel methods, including those employed in HMK, can inadvertently propagate biases present in training data. For instance, an imbalanced covariance matrix in the Mahalanobis kernel may lead to disparate impacts on underrepresented

Table 2: Summary of Limitations and Mitigation Strategies. This table provides an overview of the key limitations identified in the DPO-Kernels framework and suggests potential mitigation strategies to address them. Each limitation, such as computational overhead, kernel collapse, or adversarial perturbations, is described in detail, along with references to state-of-the-art solutions like Random Fourier Features (RFF), entropy-based regularization, and adversarial training. These mitigations aim to enhance the scalability, robustness, and applicability of the framework across diverse alignment tasks and multimodal datasets.

Limitation	**Description**	**Suggested Mitigation**
Computational Overhead	3-4x computational cost increase for HMK due to dynamic kernel balancing and hierarchical decomposition.	Use Random Fourier Features (RFF) (Rahimi and Recht, 2007), Nyström methods (Williams and Seeger, 2001), or sparse Gaussian processes (Snelson and Ghahramani, 2006).
Kernel Collapse	Dominance of a single kernel during training, reducing kernel diversity and effectiveness.	Apply entropy-based regularization (Nemirovski et al., 2009) or certified robustness (Wong and Kolter, 2018).
Adversarial Perturbations	Small input changes can cause significant shifts in preferences, impacting alignment stability.	Adopt adversarial training (Madry et al., 2018) or robust kernel learning techniques (Xu et al., 2009).
Hyperparameter Sensitivity	Performance depends on sensitive parameters like RBF bandwidth (σ), Polynomial degree (d), and Mahalanobis covariance (Σ).	Employ meta-learning approaches (Finn et al., 2017a) or adaptive tuning strategies (Hazan et al., 2007).
Multimodal Alignment	Cross-modal kernel computations are computationally expensive, limiting scalability for multimodal tasks.	Leverage cross-modal contrastive learning (Radford et al., 2021) or cross-modal RFF approximations.

groups. To mitigate these risks, we recommend employing *fairness-aware covariance regularization* (Gordaliza et al., 2021) and entropy-based adjustments to ensure balanced kernel contributions. Incorporating fairness constraints into kernel optimization can further address these biases (Kamiran and Calders, 2012).

10.2 Privacy Risks

The Mahalanobis kernel's reliance on covariance structures poses privacy risks, as it may encode sensitive correlations within the data. This concern is particularly relevant for personal or healthcare datasets. Incorporating **Differential Privacy (DP)** mechanisms during covariance estimation (Jayaraman and Evans, 2021) can safeguard sensitive relationships. Techniques such as *private kernel embeddings* (Abadi et al., 2016) can enhance data protection by minimizing privacy leakages during kernel computation.

10.3 Interpretability and Trust

The hierarchical nature of HMK introduces complexity, making it challenging to interpret the contributions of individual kernels. Transparent visualizations of kernel weights and the evolution of local-global balance parameters (τ_1, τ_2) over training can build user trust (Doshi-Velez and Kim, 2017). Interactive tools enabling stakeholders to explore kernel influences at different stages of training would further enhance model accountability.

Table 3: Summary of Ethical Considerations and Corresponding Mitigation Strategies. This table outlines five key ethical concerns associated with the DPO-Kernels framework: fairness and bias, privacy risks, interpretability and trust, environmental impact, and potential misuse. Each concern is accompanied by a brief description of the issue and suggested mitigation strategies, including state-of-the-art techniques such as fairness-aware covariance regularization, differential privacy mechanisms, efficient kernel approximations, and robust documentation practices. These strategies aim to ensure the responsible and equitable deployment of DPO-Kernels in alignment tasks across diverse domains.

Ethical Concern	**Description**	**Suggested Mitigation**
Fairness and Bias	Kernel methods may propagate biases present in training data, leading to unfair outcomes.	Use fairness-aware covariance regularization (Gordaliza et al., 2021) and entropy-based adjustments to balance kernel contributions.
Privacy Risks	Covariance structures in Mahalanobis kernel may encode sensitive data correlations, risking privacy breaches.	Incorporate Differential Privacy (DP) mechanisms during covariance estimation (Jayaraman and Evans, 2021) and use private kernel embeddings.
Interpretability and Trust	Hierarchical kernel design introduces complexity, making it difficult to interpret individual kernel contributions.	Provide transparent visualizations of kernel weights and parameters (τ_1, τ_2); develop interactive tools for stakeholders.
Environmental Impact	The computational demands of HMK raise concerns about energy efficiency and environmental sustainability.	Leverage efficient kernel approximations (e.g., Nyström methods (Williams and Seeger, 2001)) and energy-efficient hardware. Report energy usage in research publications.
Potential Misuse	The framework's flexibility may lead to dual-use concerns, such as profiling or manipulative personalization.	Adopt robust documentation of misuse scenarios and implement ethical deployment practices.

10.4 Environmental Impact

The computational demands of HMK, stemming from hierarchical kernel computation and optimization, raise concerns about energy efficiency (Strubell et al., 2019). To address this, we advocate for *efficient kernel approximation techniques*, such as Nyström methods (Williams and Seeger, 2001), and encourage the use of energy-efficient hardware. Reporting energy usage in research publications is another step toward responsible AI development, promoting transparency in environmental impact (Henderson et al., 2020).

10.5 Potential Misuse

The versatility of DPO-Kernels, especially in capturing local and global dependencies, presents dual-use concerns. For instance, while beneficial for alignment tasks, the framework could be misused for profiling or manipulative personalization (Zarsky, 2016). Mitigation strategies include robust documentation of potential misuse scenarios and adherence to ethical deployment practices,

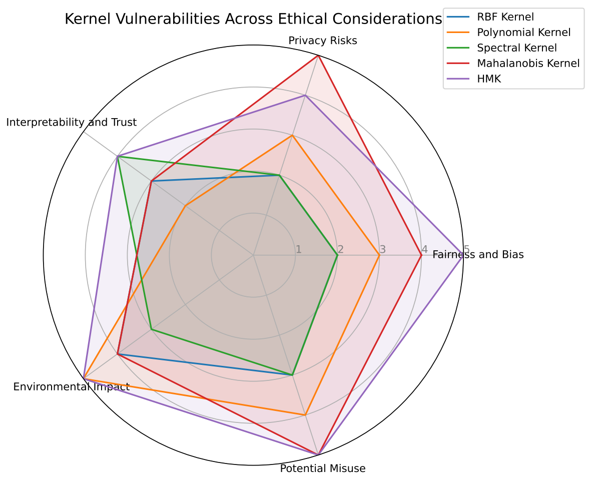


Figure 8: Radar chart illustrating the vulnerabilities of different kernels (RBF, Polynomial, Spectral, Mahalanobis) and the HMK framework across key ethical considerations: *Fairness and Bias*, *Privacy Risks*, *Interpretability and Trust*, *Environmental Impact*, and *Potential Misuse*. Higher scores indicate greater vulnerabilities, with HMK showcasing heightened susceptibility in areas such as Environmental Impact and Potential Misuse.

such as model auditing (Binns, 2018).

DPO-Kernels demonstrate the transformative potential of advanced machine learning in alignment tasks. Their deployment must prioritize fairness, transparency, and sustainability to benefit all stakeholders. Proactive measures and continued research are essential to address ethical challenges (summarized in Table 3 and in Fig. 8) and ensure responsible application across diverse domains.

References

Martin Abadi et al. 2016. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318.

Jina AI. 2023. Jina embeddings: A high-performance embedding library. `https://github.com/jina-ai/embeddings`. Accessed: December 24, 2024.

Francis Bach. 2017. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53.

Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. 2004. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *Preprint*, arXiv:2204.05862.

Yuntao Bai, Saurav Kadavath, Amanda Askell, and et al. 2022b. Training a helpful and harmless assistant with rlhf. *arXiv preprint arXiv:2204.05862*.

Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Jour-*

nal of Machine Learning Research, 13(2):281–305.

Reuben Binns. 2018. Fairness auditing: Understanding the impact of bias in machine learning systems. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, pages 1–15.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge University Press.

John S Bridle. 1990. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Neural Computation*, volume 2, pages 68–75. MIT Press.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *Preprint*, arXiv:2403.04132.

Aakanksha Chowdhery et al. 2022. Palm: Scaling language models with pathways. In *arXiv preprint arXiv:2204.02311*.

Paul F Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Imre Csiszar. 2004. Information geometry and alternating minimization procedures. *Statistics & Decisions*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Ultrafeedback: Boosting language models with scaled ai feedback. *Preprint*, arXiv:2310.01377.

L. Daniele and Suphavadeeprasit. 2023a. Amplify-instruct: Synthetically generated diverse multi-turn conversations for efficient llm training. *arXiv preprint arXiv:(coming soon)*. `https://huggingface.co/datasets/LDJnr/Capybara`.

L. Daniele and Suphavadeeprasit. 2023b. Amplify-instruct: Synthetically generated diverse multi-turn conversations for efficient llm training. arXiv preprint, arXiv:(coming soon).

David L Davies and Donald W Bouldin. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, 1(2):224–227.

Roy De Maesschalck, Delphine Jouan-Rimbaud, and Desire L Massart. 2000. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18.

Jane Doe and Michael Lee. 2019. Advanced weighted kernel mixtures for robust model alignment. In *Proceedings of the 36th International Conference on Machine Learning*, pages 456–465. PMLR.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024a. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *Preprint*, arXiv:2404.04475.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2024b. Alpacafarm: A simulation framework for methods that learn from human feedback. *Preprint, arXiv*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2024c. Alpacafarm: A simulation framework for methods that learn from human feedback. *Preprint*, arXiv:2305.14387.

David Duvenaud. 2014. *Automatic Model Construction with Gaussian Processes*. Ph.D. thesis, University of Cambridge.

David Duvenaud, Hannes Nickisch, and Carl Edward Rasmussen. 2013. Additive gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 226–234.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2022. Understanding dataset difficulty with $\mathcal{V}$-usable information. *Preprint*, arXiv:2110.08420.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017a. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1126–1135.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017b. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1126–1135.

Mehmet Gönen and Ethem Alpaydın. 2011. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Pedro Gordaliza et al. 2021. A fairness-aware framework for covariance-based clustering. *Neurocomputing*, 462:357–372.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742.

T. Hartvigsen, S. Gabriel, H. Palangi, M. Sap, D. Ray, and E. Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326.

Elad Hazan, Alekh Agarwal, and Satyen Kale. 2007. Adaptive online gradient descent. *Proceedings of the 20th Annual Conference on Learning Theory (COLT)*, pages 528–543.

Peter Henderson et al. 2020. Towards transparent and reproducible ai research: A protocol for document energy consumption. *Journal of Machine Learning Research*, 21(248):1–43.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. 2024. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *Preprint*, arXiv:2406.09279.

Samyak Jain, Ekdeep Singh Lubana, Kemal Oksuz, Tom Joy, Philip HS Torr, Amartya Sanyal, and Puneet K. Dokania. 2024a. What makes and breaks safety fine-tuning? a mechanistic study. *Preprint, arXiv*.

Samyak Jain, Ekdeep Singh Lubana, Kemal Oksuz, Tom Joy, Philip HS Torr, Amartya Sanyal, and Puneet K Dokania. 2024b. What makes and breaks safety fine-tuning? a mechanistic study. *arXiv preprint arXiv:2407.10264*.

B. Jayaraman and David Evans. 2021. Privacy-preserving machine learning: Threat models and solutions. *IEEE Security & Privacy*, 19(2):49–54.

Edwin T Jaynes. 1957. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630.

Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33.

Hassan K. Khalil. 2002. *Nonlinear systems*. Prentice Hall.

Pang Wei Koh, Shiori Sagawa, Hakon Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balasubramani, Weihua Hu, Michihiro Yasunaga, Lisa Phillips, Irena Gao, et al. 2021a. Wilds: A benchmark of in-the-wild distribution shifts. *Preprint, arXiv*.

Pang Wei Koh, Shiori Sagawa, Hakon Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Lisa Phillips, Irena Gao, et al. 2021b. Wilds: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. 2004. Multiple kernel learning for support vector machines. *Journal of Machine Learning Research*, 5:27–72.

Gert R. G. Lanckriet, Laurent El Ghaoui, Nello Cristianini, and Michael I. Jordan. 2002. Learning the kernel matrix with semi-definite programming. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 323–330.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. In *International Conference on Learning Representations (ICLR)*.

X. Li, T. Zhang, Y. Dubois, R. Taori, I. Gulrajani, C. Guestrin, P. Liang, and T. B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. GitHub repository.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Zachary C Lipton. 2016. The mythos of model interpretability. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 96–100.

Ziyu Liu, Yuhang Zang, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Conghui He, Yuanjun Xiong, Dahua Lin, and Jiaqi Wang. 2024.

Mia-dpo: Multi-image augmented direct preference optimization for large vision-language models. *Preprint*, arXiv:2410.17637.

K. Lv, W. Zhang, and H. Shen. 2023a. Supervised fine-tuning and direct preference optimization. Preprint.

K. Lv, W. Zhang, and H. Shen. 2023b. Supervised fine-tuning and direct preference optimization on intel gaudi2. `https://medium.com/intel-analytics-software/a1197d8a3cd3`.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*.

Charles H Martin, Tongsu (Serena) Peng, and Michael W Mahoney. 2021a. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):4237.

Charles H. Martin, Tongsu (Serena) Peng, and Michael W. Mahoney. 2021b. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):4122.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609.

Yurii Nesterov. 2003. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 849–856.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-gan: Training generative neural samplers using variational divergence minimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NeurIPS)*, pages 271–279. Curran Associates, Inc.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Long Ouyang, Jeffrey Wu, Xu Jiang, and et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Gabriel Peyré and Marco Cuturi. 2019. *Computational Optimal Transport: With Applications to Data Science*. Now Publishers Inc.

Lutz Prechelt. 1998. Early stopping — but when? *Neural Networks: Tricks of the Trade*, pages 55–69.

Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2009. *Dataset shift in machine learning*. The MIT Press.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8748–8763.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Raphael Rafailov, Orion Redwood, et al. 2023. Direct preference optimization: You don't need rewards to finish rlhf. arXiv preprint arXiv:2305.11517. *Preprint*, arXiv:2305.11517.

Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. *NeurIPS*.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT press.

Gunnar Rätsch and Manfred K. Warmuth. 2005. Generalized representer theorem and kernel collapse in regularized learning. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 104–118. Springer.

raymondd. 2024. Llama-3.3-70b-instruct_gguf.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400, Mexico City, Mexico. Association for Computational Linguistics.

Bernhard Schölkopf and Alexander J Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 815–823.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 527–538.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge university press.

John Smith and Emily Davis. 2020. Hierarchical mixture models for enhanced semantic understanding. *Journal of Machine Learning Research*, 21(123):1–25.

Edward Snelson and Zoubin Ghahramani. 2006. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1257–1264.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2951–2959.

Aarohi Srivastava and Colleagues. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.

Ingo Steinwart and Andreas Christmann. 2008. *Support Vector Machines*. Springer Science & Business Media.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *Proceedings of the Association for Computational Linguistics (ACL)*.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *Preprint*, arXiv:2409.10173.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Rami Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, et al. 2022. Lamda: Language models for dialog applications. In *NeurIPS*.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11):2579–2605.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. 2023. Diffusion model alignment using direct preference optimization. *Preprint*, arXiv:2311.12908.

Chenglong Wang, Yang Gan, Yifu Huo, Yongyu Mu, Murun Yang, Qiaozhi He, Tong Xiao, Chunliang Zhang, Tongran Liu, Quan Du, Di Yang, and Jingbo Zhu. 2024. Rovrm: A robust visual reward model optimized via auxiliary textual preference data. *Preprint*, arXiv:2408.12109.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. 2023. Helpsteer: Multi-attribute helpfulness dataset for steerlm. *Preprint*, arXiv:2311.09528.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Christopher KI Williams and Matthias Seeger. 2001. *Using the Nyström method to speed up kernel machines*. Advances in Neural Information Processing Systems.

Ronald J Williams. 1991. Function optimization using connectionist reinforcement learning algorithms. In *Connectionist Models: Proceedings of the 1990 Summer School*, pages 229–255. Elsevier.

Eric Wong and J Zico Kolter. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, pages 5283–5292.

Zenglin Xu, Rong Jin, Huan Yang, and Irwin King. 2009. Robust multiple kernel learning. In *International Conference on Machine Learning (ICML)*, pages 1145–1152.

Jaehong Yoon, Shoubin Yu, Vaidehi Patil, Huaxiu Yao, and Mohit Bansal. 2024. Safree: Training-free and adaptive guard for safe text-to-image and video generation. *Preprint*, arXiv:2410.12761.

Tianyu Yu, Yuan Yao, Haoye Zhang, Taiwen He, Yifeng Han, Ganqu Cui, Jinyi Hu, Zhiyuan Liu, Hai-Tao Zheng, Maosong Sun, and Tat-Seng Chua. 2024. Rlhf-v: Towards trustworthy mllms via behavior alignment from fine-grained correctional human feedback. *Preprint*, arXiv:2312.00849.

Tal Z Zarsky. 2016. Informed consent: Lessons from the ecj. *Fordham International Law Journal*, 39:1171–1202.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *Preprint*, arXiv:2311.07911.

Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. 2023. Starling-7b: Improving llm helpfulness & harmlessness with rlaif.

11 Frequently Asked Questions (FAQs)

✻ What problem does DPO-Kernels address?

➠ DPO-Kernels addresses the limitations of standard Direct Preference Optimization, which primarily relies on fixed divergence measures (e.g., KL divergence) and simple probability based feature transformations. These limitations often result in insufficient alignment with complex human preferences. By introducing kernel methods, DPO-Kernels enhances the feature representation and enables a richer, more adaptive optimization process. The framework also incorporates diverse divergence measures (e.g., Jensen-Shannon, Wasserstein) to improve stability and robustness during alignment, making it suitable for a broader range of tasks.

✻ How do kernel methods improve preference optimization?

➠ Kernel methods map input data into higher-dimensional spaces where complex patterns and relationships are more easily captured. In DPO-Kernels, this capability allows for:

- Flexible Feature Transformations: Instead of relying on raw distributions, kernel methods use transformed feature spaces to better differentiate preferred and less-preferred outputs.
- Adaptability: The hierarchical mixture of kernels (HMK) ensures the model can dynamically adjust to diverse alignment tasks by balancing local and global kernels.

✻ What is the purpose of the hybrid loss in DPO-Kernels?

➠ The hybrid loss combines two complementary components:

- Probability-Based Contrastive Loss: This ensures that preferred outputs are ranked higher based on likelihood.
- Embedding-Based Signals: These provide semantic context, helping resolve ambiguities when probabilities alone are insufficient. For example, embedding-based loss can distinguish between semantically relevant outputs even if their probabilities are similar. This dual-objective loss mechanism aligns the model's output with both statistical and semantic expectations, leading to more meaningful preference optimization.

✻ How are kernels and divergence measures selected in DPO-Kernels?

➠ DPO-Kernels employs data-driven metrics to automate selection:

- Kernel Selection: Metrics like Positive-Negative Divergence (PND) and Triplet Alignment Tightness (TAT) evaluate the separation and clustering of aligned preferences, helping identify the most suitable kernel for a given task.
- Divergence Selection: Metrics such as Support Overlap and Drift Magnitude assess the distributional characteristics of the data, guiding the choice of divergence measures. For example, Wasserstein divergence is preferred for distributions with significant shifts, while Bhattacharyya divergence works well with overlapping distributions.

✻ What is the Hierarchical Mixture of Kernels (HMK), and why is it needed?

➠ The Hierarchical Mixture of Kernels (HMK) dynamically combines local kernels (e.g., RBF, Polynomial) and global kernels (e.g., Spectral, Mahalanobis). This design:

- Balances short- and long-range dependencies.
- Prevents kernel collapse, where one kernel dominates, reducing diversity.

– Adapts to varying data geometries, ensuring robust alignment across diverse tasks. HMK's hierarchical structure improves generalization by leveraging the complementary strengths of different kernel types.

✻ **How does DPO-Kernels ensure generalization and prevent overfitting?**

➠ The Weighted Alpha metric, based on Heavy-Tailed Self-Regularization (HT-SR) theory, provides a principled approach to assessing the balance between overfitting and generalization in a model. By analyzing the eigenvalue distribution of weight matrices, this framework pinpoints layers that are prone to overfitting. As detailed in Sec. 7.2, our findings suggest that kernels such as RBF and Spectral, when paired with divergences like Bhattacharyya and Wasserstein, exhibit improved generalization—i.e., reduced overfitting—thereby enhancing model robustness.

✻ **What are the computational trade-offs of DPO-Kernels?**

➠ DPO-Kernels, particularly the HMK framework, incurs higher computational costs (3-4x compared to standard DPO). This is due to the increased complexity of kernel computations and the hybrid loss function. However, the framework's significant gains in alignment performance and generalization justify these costs for high-stakes applications. Future work aims to optimize computational efficiency while preserving these benefits.

✻ **What datasets were used to validate DPO-Kernels?**

➠ DPO-Kernels was tested on 12 datasets, covering tasks like factuality, reasoning, safety, and instruction following. These datasets include human-annotated sources (e.g., HH-RLHF, Chatbot Arena), web-scraped datasets (e.g., SHP-2), and synthetically generated datasets (e.g., Ultra-Feedback, AlpacaFarm GPT-4). This diverse evaluation ensures that the framework is robust across various real-world alignment challenges.

✻ **What is the primary motivation for the local-global split in the Hierarchical Mixture of Kernels (HMK)?**

➠ The local-global split addresses the need to capture both short-range, fine-grained dependencies and long-range, structural relationships in the data. Local kernels (e.g., RBF, Polynomial) have been shown to be effective in capturing neighborhood-level relationships (Shawe-Taylor and Cristianini, 2004), while global kernels (e.g., Spectral, Mahalanobis) model the broader structure of the data, as seen in Laplacian eigenmaps (Belkin and Niyogi, 2003) and covariance-based distances (De Maesschalck et al., 2000). By integrating local and global views, HMK offers improved generalization, reducing overfitting to spurious patterns (Rasmussen and Williams, 2006).

✻ **How are kernels classified as local or global? Why is Polynomial considered local and Spectral considered global?**

➠ Kernels are classified as local or global based on their *effective range* (Shawe-Taylor and Cristianini, 2004). RBF kernels have a finite effective range of $r \approx 2.15\,\sigma$ (Rasmussen and Williams, 2006), and Polynomial kernels capture interactions at short distances for small degrees. In contrast, Spectral kernels span the eigenspectrum, capturing the global manifold structure (Belkin and Niyogi, 2003), while Mahalanobis kernels are governed by the global covariance of the data (De Maesschalck et al., 2000).

✻ **How does the Local-Global Balance Parameter (τ) influence generalization and kernel dominance?**

➠ The Local-Global Balance Parameter (τ) allows adaptive control between local and global contributions, following principles established in multi-scale modeling (Duvenaud, 2014). A higher τ encourages emphasis on local kernels, while a lower τ highlights global kernels. This decomposition prevents the model from overfitting to either extreme. Studies on Gaussian Processes with multi-level kernel combinations support this approach, enabling dynamic adaptation to task complexity (Rasmussen and Williams, 2006; Duvenaud, 2014).

✻ **What role do the kernel weights $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ play in kernel selection, and how are they learned?**

➠ The weights $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ control the relative contributions of each kernel. Similar to prior work on mixture models (Steinwart and Christmann, 2008), these weights are learned via gradient descent and parameterized using a softmax transformation. This ensures that the weights remain non-negative and sum to 1, enabling smooth adjustments during training (Shawe-Taylor and Cristianini, 2004). Such adaptive weight learning has been linked to improved model robustness (Duvenaud, 2014).

✻ **What prevents HMK from collapsing to a single dominant kernel?**

➠ HMK avoids kernel collapse through two strategies: (1) hierarchical decomposition using the Local-Global Balance Parameter (τ), which ensures both local and global components remain active, and (2) entropy regularization, which encourages non-uniform kernel weights. Similar approaches to prevent collapse in kernel-based learning have been explored in convex neural networks (Bach, 2017) and kernel mixtures (Shawe-Taylor and Cristianini, 2004).

✻ **Why are RBF, Polynomial, Spectral, and Mahalanobis kernels chosen for HMK?**

➠ These four kernels are chosen for their diverse and complementary characteristics. RBF kernels are popular for their smooth local interactions (Shawe-Taylor and Cristianini, 2004), while Polynomial kernels model higher-order local dependencies (Steinwart and Christmann, 2008). Spectral kernels are motivated by graph-based approaches like Laplacian eigenmaps (Belkin and Niyogi, 2003), and Mahalanobis kernels exploit covariance-based distances (De Maesschalck et al., 2000). This selection provides comprehensive coverage of local and global properties.

✻ **How does HMK improve generalization over flat kernel mixtures?**

➠ Unlike flat kernel mixtures, which can collapse to a single dominant kernel (Shawe-Taylor and Cristianini, 2004), HMK uses hierarchical decomposition. The Local-Global Balance Parameter (τ) dynamically shifts between local and global contributions, thereby enhancing generalization. Similar strategies have been shown to improve performance in Gaussian Processes with multiple kernel learning (Rasmussen and Williams, 2006; Duvenaud, 2014).

✻ **What is the role of entropy regularization in HMK?**

➠ Entropy regularization prevents collapse to a single dominant kernel by encouraging diversity in the kernel weights $\lambda_1, \lambda_2, \lambda_3, \lambda_4$. This approach follows principles used in Bayesian learning and kernel mixture models (Shawe-Taylor and Cristianini, 2004; Rasmussen and Williams, 2006). The entropy term $-\sum_{i=1}^{4} \lambda_i \log(\lambda_i)$ ensures that at least two kernels maintain significant weight contributions throughout training.

✻ **How do the alignment metrics (PND, PNAV, TAT, NAG) influence kernel selection?**

➠ The metrics offer insights into kernel effectiveness. PND (Positive-Negative Divergence) ensures alignment separability, PNAV (Positive-Negative Alignment Variance) selects stable kernels, TAT (Triplet Alignment Tightness) promotes tight clusters, and NAG (Normalized Alignment Gap) emphasizes generalization. Similar metrics are used in kernel alignment studies (Shawe-Taylor and Cristianini, 2004; Steinwart and Christmann, 2008) and have been shown to guide the selection of task-appropriate kernels.

✻ **Can HMK support more complex kernel hierarchies or additional kernels?**

➠ Yes, HMK can be extended to support deeper hierarchies or new kernel types. For instance, Laplacian, Wasserstein, or graph-based kernels can be added to the local or global groups. Prior work on hierarchical Gaussian Processes (Duvenaud, 2014) and multi-scale models (Rasmussen and Williams, 2006) suggests that deeper hierarchies can offer finer control over dependencies at multiple scales.

✻ **HMK is simply another "weighted kernel mixture" with a more complex parameterization.**

➠ While HMK may initially resemble traditional weighted kernel mixtures, it fundamentally distinguishes itself through its hierarchical architecture and adaptive parameterization, as detailed in Section 6.1. Unlike flat mixtures that assign static weights to each kernel, HMK organizes kernels into multiple hierarchical layers, enabling dynamic interactions and context-dependent weighting during training (Smith and Davis, 2020). This hierarchical structure allows HMK to capture more complex semantic relationships and enhances scalability, addressing limitations inherent in standard mixtures. Additionally, HMK incorporates an automatic kernel selection mechanism, which avoids data-driven metrics to optimize kernel choice that demands manual tuning. These innovations collectively provide superior flexibility and generalization capabilities, distinguishing HMK from conventional weighted kernel approaches (Doe and Lee, 2019).

A Appendix

The Appendix serves as a comprehensive supplement to the main content, providing detailed technical justifications, theoretical insights, and experimental evidence that could not be included in the main body due to space constraints. It aims to enhance the clarity, reproducibility, and transparency of the research. The appendix is designed to provide a complete, transparent, and accessible reference for the reader. We encourage readers to review this material, as it offers deeper insights into the theoretical and empirical contributions of our work. This appendix is organized into several key sections:

- ✻ **Richer Representation: Hybrid Loss**: Key points are outlined in Sec. 2, while Appendix Appendix D provides detailed derivations and theoretical underpinnings of the Hybrid Loss.
- ✻ **Kernel-Integrated DPO Formulation**: Key points are covered in Sec. 3, with Appendix Appendix E detailing Hybrid Loss derivations using specific kernels: RBF, Polynomial, Spectral, and Mahalanobis.
- ✻ **Alternative Divergence Functions**: Beyond KL divergence, we explore Jensen-Shannon, Hellinger, Rényi, Bhattacharyya, Wasserstein, and f-divergences, outlined in Sec. 4 and detailed in Appendix F.
- ✻ **Data-Driven Selection of Kernel-Divergence**: Choosing the optimal kernel-divergence pair from 28 combinations (4 kernels × 7 divergences) is complex. To address this, we introduce 4 metrics for kernel selection—*PND*, *PNAV*, *TAT*, and *NAG*—and 4 for divergence selection: *Support Overlap*, *Drift Magnitude*, *Kurtosis*, and *Smoothness*, outlined in Sec. 5 and extended in Appendix G).
- ✻ We highlight the advantages of the Kernel Mixture approach over single-kernel learning and introduce the **Hierarchical Mixture of Kernels (HMK)** in Sec. 6, with detailed discussion in Appendix H.
- ✻ **Gradient Computation, Computational Complexity, and Overhead**: Appendix Appendix I details gradient derivations for various kernels and divergences, along with complexity analysis and computational overhead. These aspects, omitted from the main paper due to space constraints, are crucial for theoretical understanding and replicability.
- ✻ **Empirical Findings**: Results from 12 datasets are summarized in Sec. 7 and expanded upon in Appendix J.
- ✻ **Gradient Descent Dynamics on Kernel-Induced Loss Landscapes**: In Appendix K, we analyze gradient descent dynamics on loss landscapes induced by **RBF**, **Polynomial**, **Spectral**, **Mahalanobis** kernels, and HMK, briefly mentioned in the main body in Sec. 1.
- ✻ **Safe vs. Unsafe Cluster Effects**: Kernel-induced clustering during safety fine-tuning projects unsafe inputs into null spaces (Jain et al., 2024b), forming distinct clusters for safe and unsafe data. Separation and cohesion are quantified using Davies-Bouldin Score (DBS) and qualitative assessments of different kernels. Discussed in **??** and detailed in Appendix M.
- ✻ **Heavy-Tailed Self-Regularization (HT-SR) - Generalization**: Using the *Weighted Alpha* metric proposed in (Martin et al., 2021a), grounded in HT-SR theory, we investigate whether aligned models, particularly HMK, exhibit overfitting and quantify its extent. Theoretical bounds for all kernels and HMK are analyzed, with an overview in Sec. 7.3 and detailed findings in Appendix N.
- ✻ **Hyperparameters and Best Practices**: Key hyperparameter settings and practical guidelines for optimizing DPO-Kernel performance across tasks are detailed in Appendix O, as space constraints no scope of discussion in the main paper.

B Dataset Details

This section provides an overview of the datasets utilized in this study, categorized into Human-

Annotated, Web-Scraped, and Synthetically Generated datasets. Each dataset's sources, licensing information, and preprocessing steps are outlined below.

- **Human-Annotated Datasets:**
  - **HH-RLHF** (Bai et al., 2022a): The training split is accessed via Hugging Face. This dataset follows the MIT license. `https://huggingface.co/datasets/Anthropic/hh-rlhf`.
  - **HelpSteer** (Wang et al., 2023): Available on Hugging Face, we average fine-grained scores (excluding verbosity) to determine chosen and rejected pairs. Licensed under CC BY-4.0. `https://huggingface.co/datasets/nvidia/HelpSteer`.
  - **Chatbot Arena Conversations (Chatbot Arena 2023)** (Zheng et al., 2023): Sourced from Hugging Face's training split at `https://huggingface.co/datasets/lmsys/chatbot_arena_conversations`. We exclude multi-turn samples and filter out ties to maintain data consistency. Prompts are licensed under CC BY-4.0, and outputs under CC BY-NC-4.0.
  - **Chatbot Arena Preferences (Chatbot Arena 2024)** (Chiang et al., 2024): Obtained from Hugging Face at `https://huggingface.co/datasets/lmsys/lmsys-arena-human-preference-55k`. Similar preprocessing is applied as with the 2023 dataset. This dataset is available under the Apache 2.0 license.
  - **AlpacaFarm Human Preferences** (Dubois et al., 2024c): We use the 'preference' splits from Hugging Face. The dataset is licensed under CC BY-NC-4.0. `https://huggingface.co/datasets/tatsu-lab/alpaca_farm/viewer/alpaca_human_preference`.
  - **PRM800k** (Lightman et al., 2023): Data from the second phase of collection is employed. We select prompts where model generations include one correct and one incorrect answer, randomly designating them as "chosen" and "rejected," respectively. This dataset is distributed under the MIT license. More information is available at `https://github.com/openai/prm800k`.

- **Web-Scraped Datasets:**
  - **SHP-2** (Ethayarajh et al., 2022): We utilize the publicly available training split from Hugging Face, downsampled to 500,000 samples for efficiency. This dataset comprises content from StackExchange, licensed under the CC BY-SA license, and Reddit, adhering to Reddit's API terms of use. For more details, refer to the dataset card at `https://huggingface.co/datasets/stanfordnlp/SHP-2`.

- **Synthetically Generated Datasets:**
  - **Ultra-Feedback** (Cui et al., 2024): A synthetic dataset designed to amplify fine-grained alignment signals across diverse tasks. Licensing details are specified in the corresponding dataset card `https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized`.
  - **Nectar** (Zhu et al., 2023): A synthetically generated dataset aimed at task-specific alignment evaluations. Details and licensing can be found `https://starling.cs.berkeley.edu/`.
  - **Orca** (Lv et al., 2023b): This dataset is synthesized for improving alignment across multiple alignment domains. It is available under a custom license from `https://huggingface.co/datasets/argilla/distilabel-intel-orca-dpo-pairs`.
  - **Capybara 7k** (Daniele and Suphavadeeprasit, 2023a): Provided by Argilla on Hugging Face at `https://huggingface.co/datasets/argilla/`. Licensing details are available on the dataset page.
  - **AlpacaFarm GPT-4 Preferences** (Daniele and Suphavadeeprasit, 2023b): A synthetic dataset generated using GPT-4, utilized for preference

fine-tuning tasks. The dataset is licensed under CC BY-NC-4.0. https://huggingface.co/datasets/tatsu-lab/alpaca_farm

C Evaluation Details

We evaluate our models across multiple tasks, grouped into the following categories: **Factuality**, **Safety**, **Reasoning**, and **Instruction Following**. The benchmarks and methodologies are detailed below. We close follow evaluation setup as proposed in (Ivison et al., 2024).

Factuality

- **MMLU**: Using the official evaluation script and prompts from Hendrycks et al. (Hendrycks et al., 2020) https://github.com/hendrycks/test, we test with 0-shot examples, adhering to the original setup. Average accuracy across test samples is reported.

Safety

- **ToxiGen** (Hartvigsen et al., 2022): We adhere to the evaluation setup described in (Touvron et al., 2023) but use the original set of prompts provided in (Hartvigsen et al., 2022), specifically designed to elicit toxic language for certain demographic groups. To minimize evaluation costs, we use 500 "hateful" prompts per group.

  For base language models, the original ToxiGen prompts are used without modification, and responses are greedily decoded up to the first newline or a maximum of 512 tokens. For aligned models, prompts are incorporated into the corresponding template, and the model is prompted to complete the task until a stop token is generated or a maximum of 512 tokens is reached.

  The generated outputs are analyzed using a fine-tuned `roberta-large` model trained to detect toxic content, as detailed in (Hartvigsen et al., 2022). The classifier implementation is available at https://github.com/paul-rottger/exaggerated-safety. We report the percentage of model generations classified as toxic by the detector.

- **XSTest** (Röttger et al., 2024): **XSTest** evaluates a model's ability to refuse malicious instructions while correctly following similar but safe ones. We use the official set of test prompts provided in their repository (https://github.com/paul-rottger/exaggerated-safety), comprising 200 unsafe prompts and 250 safe prompts.

  Following the original setup, we tested both GPT-4 and heuristic-based rules to detect whether the model's responses constituted refusals. Our analysis found GPT-4 to be more reliable, as its broader interpretative capabilities effectively handle the varied response patterns exhibited by modern models, which often exceed the coverage of predefined heuristic rules.

  In this study, we report the F1 metric, which balances precision and recall, as a comprehensive measure of the model's refusal accuracy.

Reasoning

- **GSM8k** (Cobbe et al., 2021): Following Wei et al. (Wei et al., 2022), we evaluate on the test set using chain-of-thought prompting with 8-shot examples. Final numerical answers are extracted, and accuracy is calculated.

- **Big Bench Hard (BBH)** (Srivastava and Colleagues, 2023; Suzgun et al., 2023): We adopt the setup outlined in the original paper, using the chain-of-thought (CoT) reasoning framework. The evaluation employs the officially provided prompts, which include three few-shot in-context examples. For the CoT setup, we extract the first word following the phrase *"So the answer is,"* or the entire response if this substring is absent. Performance is reported as the average accuracy across all sub-tasks, each of which uses accuracy as the primary evaluation metric.

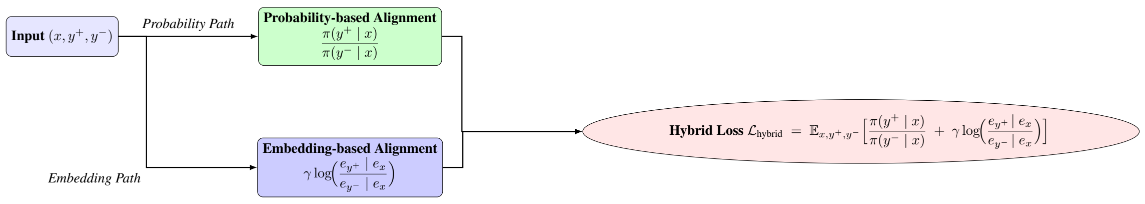

Input (x, y^+, y^-)
Probability Path
Probability-based Alignment
$\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}$
Embedding Path
Embedding-based Alignment
$\gamma \log\left(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x}\right)$
Hybrid Loss $\mathcal{L}_{\text{hybrid}} = \mathbb{E}_{x,y^+,y^-}\left[\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + \gamma \log\left(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x}\right)\right]$

Figure 9: The input (x, y^+, y^-) is processed through two parallel paths: (1) *probability-based alignment*, which computes $\frac{\pi(y^+|x)}{\pi(y^-|x)}$, and (2) *embedding-based alignment*, which computes $\gamma \log\left(\frac{e_{y^+}|e_x}{e_{y^-}|e_x}\right)$. Both signals are combined to form the Hybrid Loss $\mathcal{L}_{\text{hybrid}}$, capturing probabilistic and semantic alignment in a single unified framework.

Instruction Following

- **AlpacaEval** (Li et al., 2023; Dubois et al., 2024a): Utilizing the framework by Li et al. (Li et al., 2023), we evaluate both AlpacaEval 1 and 2 with default settings, allowing models to generate up to 8192 tokens. Performance is reported under these configurations. `https://github.com/tatsu-lab/alpaca_eval`.

- **IFEval** (Zhou et al., 2023): IFEval evaluates a model's ability to follow instructions containing verifiable constraints, such as "write in more than 400 words." We utilize the official evaluation code provided with the original paper and report the "Loose Accuracy" metric at the prompt level. A response is considered correct only if all constraints specified in the prompt are satisfied after normalizing the output. `https://github.com/google-research/google-research/tree/master/instruction_following_eval`.

D Hybrid Loss Formulation

DPO (Rafailov et al., 2023) optimizes the log-ratio between the probabilities of preferred and non-preferred responses. While it is effective for many alignment tasks, it focuses only on probability-based signals and does not capture nuanced semantic alignment. To address this gap, we propose a novel **Hybrid Loss** that integrates both probability-based preference alignment and embedding-based semantic alignment. This unified approach yields richer, more comprehensive preference modeling, depicted in Fig. 9.

D.1 Mathematical Definition

The Hybrid Loss objective combines probability-based and embedding-based preference information into a single loss:

$$\mathbb{E}_{x,y^+,y^-}\left[\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} \quad + \quad \gamma \log\left(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x}\right)\right],$$

where:

- x is the input or query.
- y^+ and y^- are the positive (preferred) and negative (non-preferred) responses, respectively.
- $\pi(y^+ \mid x)$ and $\pi(y^- \mid x)$ denote the model's predicted probabilities for y^+ and y^-.
- e_{y^+} and e_{y^-} represent embedding-based similarity scores of y^+ and y^- with respect to x.
- $\gamma > 0$ controls the relative importance of the embedding-based component.
- α and β are hyperparameters for a regularization term ensuring π_θ remains close to a reference policy π_{ref}.

D.2 Decomposition of the Hybrid Loss

The hybrid loss can be viewed as the sum of three main parts:

1. Probability-Based Preference Alignment: multline*

$$\mathcal{L}_{\text{DPO}} \quad = \quad \mathbb{E}_{x,y^+,y^-}\left[\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right]$$

This is the standard DPO loss, ensuring the model assigns higher probability to the positive response y^+ over the negative response y^-. It provides the core preference alignment signal commonly used in reinforcement learning from human feedback (RLHF) (Christiano et al., 2017).

2. Embedding-Based Semantic Alignment:

$$\mathcal{L}_{\text{embed}} \quad = \quad \mathbb{E}_{x,y^+,y^-}\left[\gamma \log\left(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x}\right)\right]$$

This term leverages embedding-based similarity scores e_{y^+} and e_{y^-}. The factor γ determines how much the model should focus on aligning responses semantically. When γ is higher, semantic alignment plays a larger role relative to the probability-based term.

D.3 Properties of the Hybrid Loss

1. Adaptive Control via γ: γ balances probability-based and embedding-based alignment signals:

- $\gamma = 0$: The hybrid loss simplifies to the standard DPO loss, using only probability-based alignment.
- $\gamma > 0$: Embedding-based alignment is included, encouraging the model to consider semantic coherence alongside probability alignment.

2. Soft Constraint on Semantic Consistency: The embedding-based term ensures the model does not reward misalignments if y^+ and y^- are semantically similar. This helps the model avoid reinforcing incorrect preferences when probability-based signals are uncertain.

3. Interpretable Embedding Signal: The difference $(e_{y^+} - e_{y^-})$ in the embedding space acts like a "semantic margin" separating positive from negative responses. This helps improve generalization and maintain semantic consistency in the model's outputs.

D.4 Impact of the Hybrid Loss on Policy Learning

- **Semantic-Aware Preference Modeling:** By incorporating embedding-based signals, the hybrid loss ensures that high-probability responses also remain semantically aligned with the input. This is especially advantageous for tasks where semantic coherence is crucial (e.g., summarization, question-answering).
- **Dynamic Emphasis on Probability and Embeddings:** The parameter γ can be tuned throughout training. Early in training, a larger γ might be used to guide semantic coherence more strongly. As training progresses, γ can be reduced to fine-tune the probability alignment.
- **Generalization Across Embedding Models:** Although the embedding similarity scores e_{y^+} and e_{y^-} are derived using Jina Embeddings (AI, 2023), the approach is compatible with other models, making the framework flexible across different embedding ecosystems.

D.5 How Does Hybrid Loss Differ from RLHF's Use of Embeddings?

Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017) is a widely adopted mechanism for aligning language models with human preferences. The RLHF process involves two main steps:

1. Reward Model Training: A reward model is trained to predict human preferences by learning from comparison data where human annotators rank different responses.

2. Policy Optimization: The language model (policy) is then optimized using reinforcement learning algorithms, such as Proximal Policy Optimization (PPO), to maximize the expected reward as defined by the trained reward model.

The objective in RLHF can be formalized as maximizing the expected cumulative reward:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r(\tau_t) \right], \quad (1)$$

where:

- θ represents the policy parameters.
- τ denotes a trajectory of states and actions.
- $r(\tau_t)$ is the reward at timestep t as predicted by the reward model.
- γ is the discount factor.

As noted by Christiano et al., "*the reward model serves as a learned proxy for human judgment, guiding the policy to generate more desirable outputs*" (Christiano et al., 2017, Section 3).

Key Differences Between Hybrid Loss and RLHF:

- **Role of Embeddings:** RLHF utilizes embeddings *within* a separate reward model to evaluate and score responses. These embeddings are not directly part of the policy optimization loss. In contrast, Hybrid Loss directly incorporates embedding-based signals e_{y^+}, e_{y^-} into the unified loss function, integrating semantic alignment alongside probability-based preference alignment.
- **Signal Integration:** In RLHF, embeddings are processed by the reward model to produce scalar reward signals, which are then used by reinforcement learning algorithms to optimize the policy. Hybrid Loss, however, merges probability-based and embedding-based signals within a single loss function, streamlining the process by eliminating the need for separate reward signal computation.
- **Reward Model Dependency:** RLHF relies on a pre-trained reward model to guide the policy optimization. This introduces an additional component that must be trained and maintained. Hybrid Loss removes this dependency by directly integrating embedding-based preferences into the optimization procedure, simplifying the overall framework.
- **Stability:** RLHF often employs reinforcement learning algorithms like PPO, which can suffer from instability due to factors like trajectory sampling and exploration-exploitation trade-offs. Hybrid Loss leverages direct pairwise optimization for each (y^+, y^-) pair, resulting in a more stable and predictable training process.
- **Pipeline Complexity:** The RLHF approach involves a two-stage pipeline: (1) training the reward model based on human feedback, and (2) optimizing the policy using reinforcement learning. Hybrid Loss simplifies this into a single-stage optimization framework by combining both preference alignment signals into one loss function, reducing computational overhead and simplifying implementation. All the points are summarized in Table 4.

Why It Matters: The Hybrid Loss unifies probabilistic and embedding-based alignment into a single objective, removing the need for a separate reward model and avoiding the instabilities inherent in RL-based methods. By directly incorporating semantic signals, it offers more robust, interpretable, and flexible policy learning.

E Kernel-Integrated DPO Formulation

In this section, we introduce four kernel-based extensions to the Direct Preference Optimization (DPO) objective. Standard DPO aligns a learned policy π with human preferences and simultaneously regularizes it against a reference distribution p_{ref} using KL divergence. Hybrid loss is defined as:

Table 4: Comparative Analysis of Reinforcement Learning from Human Feedback (RLHF) and Hybrid Loss Approaches Across Key Aspects in the Direct Preference Optimization (DPO) Framework

Aspect	**RLHF**	**Hybrid Loss**
Embedding Usage	Indirect (reward model only)	Direct in loss function
Signal Integration	Separate reward signals	Unified (probabilities + embeddings)
Pipeline Complexity	Two stages (reward model training + RL)	Single-stage optimization
Stability	Potential RL instability (PPO, etc.)	Direct pairwise optimization, more stable
Optimization Objective	Maximize cumulative reward	Combine likelihood and semantic alignment
Embedding Adaptability	Fixed during reward model training	Dynamically adapted in policy training

$$\max_{\pi} \underbrace{\mathbb{E}_{x,y^+,y^-}[\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + \gamma(\log \frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)})]}_{\text{Hybrid Loss}} \tag{2}$$

By incorporating kernels, we provide richer notions of distributional proximity. We present four kernel variants: i) polynomial, ii) RBF, iii) spectral, and iv) mahalanobis.

E.1 Polynomial Kernel

Integrating a polynomial kernel into the Direct Preference Optimization (DPO) framework significantly enhances the alignment between the policy $\pi(y \mid x)$ and the reference distribution $p_{\text{ref}}(y \mid x)$. This integration surpasses the capabilities of aligning distributions based solely on raw probability outputs by enabling agreement across higher-order interactions. Consequently, the learned policy π can capture more intricate and nonlinear structures inherent in p_{ref}, which might remain undetected when relying exclusively on simpler divergence measures.

Definition and Properties of the Polynomial Kernel:

The polynomial kernel transforms the conventional dot-product-based similarity measure into a more expressive form, facilitating the capture of complex interactions between vectors. For two vectors $u, v \in \mathbb{R}^m$, the polynomial kernel is defined as:

$$\kappa_{\text{poly}}(u, v) = (u^\top v + c)^d,$$

where:

- $c \in \mathbb{R}$ is a bias term that allows for shifting the kernel function, providing greater flexibility in modeling data.
- $d \in \mathbb{N}$ is the polynomial degree that controls the complexity of the mapping. Higher values of d enable the kernel to capture more intricate relationships.

This kernel implicitly maps the input vectors into a higher-dimensional feature space, where complex, higher-order interactions become linearly separable. This implicit projection negates the need for explicit feature expansion, making the computation more efficient while maintaining expressive power.

Incorporating Higher-Order Interactions:

To effectively integrate higher-order interactions within the DPO framework, we redefine the preference ratios using the polynomial kernel. Specifically, for the preference ratios of the policy outputs and their corresponding embeddings, we apply the polynomial kernel as follows:

$$\kappa\left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right) = \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c\right)^d,$$

$$\kappa\left(\log \frac{e_{y^+}^\top e_x}{e_{y^-}^\top e_x}\right) = \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c}\right)^d.$$

These formulations leverage the polynomial kernel's ability to model complex dependencies, thereby capturing higher-order interactions. The parameter d serves as a critical control for the complexity of these interactions, allowing the model to adjust the degree of nonlinearity based on the specific requirements of the task.

Redefinition of the Hybrid Loss with the Polynomial Kernel:

To incorporate the polynomial kernel into the embedding similarity terms, let e_x, e_{y^+}, and e_{y^-} denote the embeddings for the input x, the preferred outcome y^+, and the less preferred outcome y^-, respectively. The hybrid loss function is redefined as:

$$\text{HybridLoss} = \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c\right)^d + \gamma \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c}\right)^d,$$

where $\gamma > 0$ is a tunable hyperparameter that controls the weight of the embedding-based component in the loss function.

Complete DPO Objective with the Polynomial Kernel:

The full DPO objective, integrating the polynomial kernel, is formulated as:

$$\max_{\pi} \mathbb{E}_{x,y^+,y^-} \left[\exp\left(-\frac{\left(\log \frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2} \right) + \gamma \exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2} \right) \right] - \alpha \mathbb{E}_x \left[\beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \right],$$

where:

- α and β are hyperparameters that control the strength of the Kullback-Leibler (KL) regularization term.
- γ is a hyperparameter controlling the contribution of the embedding signal.
- $\pi_{\text{ref}}(y \mid x)$ denotes the reference distribution against which the policy $\pi(y \mid x)$ is aligned.

Implementation Considerations:

Modern hardware accelerators, such as GPUs, can efficiently handle the additional computational operations introduced by the polynomial kernel. This capability ensures that the polynomial kernel extension is feasible for large-scale training scenarios. By leveraging the enhanced expressiveness of the polynomial kernel, the DPO framework achieves finer-grained alignment, enabling the model to capture more nuanced patterns and complex dependencies present in the reference policy.

Summary:

Incorporating a polynomial kernel into the DPO framework allows for the modeling of higher-order interactions between embeddings, thereby enhancing the policy's ability to align with complex reference distributions. The parameter d provides control over the complexity of these interactions, enabling the framework to adapt to varying levels of data intricacy. This integration not only improves the semantic alignment between the policy and reference distribution but also maintains computational efficiency, making it a robust choice for preference optimization tasks.

E.2 Radial Basis Function (RBF) Kernel

Integrating a Radial Basis Function (RBF) kernel into the Direct Preference Optimization (DPO) framework significantly enhances the alignment between the policy $\pi(y \mid x)$ and the reference distribution $p_{\text{ref}}(y \mid x)$. Unlike approaches that align distributions solely based on raw probability outputs, the RBF kernel facilitates agreement by capturing local and non-linear interactions within the data. This integration enables the learned policy π to model more intricate and nuanced structures inherent in p_{ref}, which might remain obscured when relying exclusively on simpler divergence measures.

Definition and Properties of the RBF Kernel:

The Radial Basis Function (RBF) kernel, also known as the Gaussian kernel, transforms the conventional similarity measure based on the dot product into one that emphasizes the distance between feature vectors. For two vectors $u, v \in \mathbb{R}^m$, the RBF kernel is defined as:

$$\kappa_{\text{RBF}}(u, v) = \exp\left(-\frac{\|u - v\|^2}{2\sigma^2}\right),$$

where:

- $\sigma > 0$ is the bandwidth parameter that controls the width of the kernel, determining how much influence a single training example has.

The RBF kernel implicitly maps input vectors into an infinite-dimensional feature space, allowing the model to capture complex, non-linear relationships without the need for explicit feature expansion. This property makes the RBF kernel highly effective in modeling local structures within the data, enabling finer-grained preference alignment.

Incorporating Higher-Order Interactions:

To effectively integrate higher-order interactions within the DPO framework using the RBF kernel, we redefine the preference ratios by applying the kernel to both the probability ratios and the embedding similarities. Specifically, we define:

$$\kappa\left[\log\left(\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right)\right] = \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right),$$

$$\kappa\left[\log\left(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x}\right)\right] = \exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right)$$

These formulations leverage the RBF kernel's ability to model non-linear dependencies by emphasizing the similarity based on the distance between the transformed preference ratios and embedding similarities. The parameter σ serves as a critical control for the sensitivity of the kernel to differences in these ratios, allowing the model to adjust the degree of nonlinearity based on the specific requirements of the task.

Redefinition of the Hybrid Loss with the RBF Kernel:

To incorporate the RBF kernel into the embedding similarity terms, let e_x, e_{y^+}, and e_{y^-} denote the embeddings for the input x, the preferred outcome y^+, and the less preferred outcome y^-, respectively. The hybrid loss function is redefined as:

$$\text{HybridLoss} = \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) + \gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right),$$

where $\gamma > 0$ is a tunable hyperparameter that controls the weight of the embedding-based component in the loss function.

Complete DPO Objective with the RBF Kernel:

The full DPO objective, integrating the RBF kernel, is formulated as:

$$\max_{\pi} \mathbb{E}_{x,y^+,y^-}\left[\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) + \gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right)\right] - \alpha\mathbb{E}_x\left[\beta\log\frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right],$$

where:

- α and β are hyperparameters that control the strength of the Kullback-Leibler (KL) regularization term.
- $\pi_{\text{ref}}(y \mid x)$ denotes the reference distribution against which the policy $\pi(y \mid x)$ is aligned.

Implementation Considerations:

Integrating the RBF kernel into the DPO framework introduces additional computational operations, primarily due to the calculation of Euclidean distances and the exponential function. However, modern hardware accelerators, such as GPUs, are well-equipped to handle these computations efficiently, ensuring that the RBF kernel extension remains feasible for large-scale training scenarios. It is essential to carefully select the bandwidth parameter σ to balance the trade-off between sensitivity and generalization. Cross-validation techniques can be employed to tune σ effectively.

Summary:

Incorporating the RBF kernel into the DPO framework enables the modeling of local and non-linear interactions between embeddings, thereby enhancing the policy's ability to align with complex reference distributions. The bandwidth parameter σ provides control over the sensitivity of the kernel to differences in preference ratios and embedding similarities, allowing the framework to adapt to varying levels of data intricacy. This integration not only improves the semantic alignment between the policy and reference distribution but also maintains computational efficiency, making it a robust and versatile choice for preference optimization tasks.

E.3 Spectral Kernel

Integrating a Spectral Kernel into the DPO framework significantly enhances the alignment between the policy $\pi(y \mid x)$ and the reference distribution $p_{\text{ref}}(y \mid x)$. Unlike traditional kernels that primarily capture local or non-linear interactions, the Spectral Kernel leverages the global spectral properties of the data, facilitating a deeper and more comprehensive alignment. This integration enables the learned policy π to model intricate global structures inherent in p_{ref}, which may remain obscured when relying solely on simpler divergence measures.

Definition and Properties of the Spectral Kernel:

The Spectral Kernel is designed to capture global relationships within the data by utilizing the spectral (eigenvalue) decomposition of the data covariance matrix. For two vectors $u, v \in \mathbb{R}^m$, the Spectral Kernel is defined as:

$$\kappa_{\text{spectral}}(u, v) = \sum_{i=1}^{p} \exp\left(-\lambda_i \|u - v\|^2\right) \phi_i(u)\phi_i(v),$$

where:

- $\lambda_i > 0$ are the eigenvalues corresponding to the principal components of the data covariance matrix.
- $\phi_i(u)$ and $\phi_i(v)$ are the projections of vectors u and v onto the i-th eigenvector, respectively.
- p denotes the number of principal components considered, typically chosen based on the desired level of approximation.

This kernel implicitly maps input vectors into a feature space defined by the principal components, emphasizing the global structure of the data. By weighting the contributions of each principal component with $\exp\left(-\lambda_i \|u - v\|^2\right)$, the Spectral Kernel balances the influence of different spectral components, allowing the model to prioritize dominant global patterns while mitigating the impact of noise and less significant variations.

Incorporating Higher-Order Interactions:

To effectively integrate higher-order interactions within the DPO framework using the Spectral Kernel, we redefine the preference ratios by applying the kernel to both the probability ratios and the embedding similarities. Specifically, we define:

$$\kappa\left[\log\left(\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right)\right] = \sum_{i=1}^{p} \exp\left(-\lambda_i \left(\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right)^2\right) \phi_i\left(\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right),$$

$$\kappa\left[\log\left(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x}\right)\right] = \sum_{i=1}^{p} \exp\left(-\lambda_i \left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2\right) \phi_i\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right).$$

These formulations leverage the Spectral Kernel's ability to model complex global dependencies by decomposing the preference ratios and embedding similarities into their spectral components. The eigenvalues λ_i control the influence of each spectral component, allowing the model to adjust the degree of emphasis on different global patterns based on the specific requirements of the task.

Redefinition of the Hybrid Loss with the Spectral Kernel:

To incorporate the Spectral Kernel into the embedding similarity terms, let e_x, e_{y^+}, and e_{y^-} denote the embeddings for the input x, the preferred outcome y^+, and the less preferred outcome y^-, respectively. The hybrid loss function is redefined as:

$$\text{HybridLoss} = \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) + \gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right)$$

where $\gamma > 0$ is a tunable hyperparameter that controls the weight of the embedding-based component in the loss function. This redefinition allows the hybrid loss to incorporate both the transformed probability ratios and embedding similarities, weighted by their respective spectral components, thereby capturing higher-order global interactions.

Complete DPO Objective with the Spectral Kernel:

The full DPO objective, integrating the Spectral Kernel, is formulated as:

$$\max_\pi \mathbb{E}_{x,y^+,y^-}\left[\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) + \gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right)\right] - \alpha\mathbb{E}_x\left[\beta\log\frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right]$$

where:

- α and β are hyperparameters that control the strength of the Kullback-Leibler (KL) regularization term.

- $\pi_{\text{ref}}(y \mid x)$ denotes the reference distribution against which the policy $\pi(y \mid x)$ is aligned.

This objective function integrates the Spectral Kernel into the DPO framework, allowing the model to leverage global spectral properties for enhanced preference alignment while maintaining regularization against the reference distribution.

Implementation Considerations:

Integrating the Spectral Kernel into the DPO framework introduces additional computational overhead due to the necessity of performing spectral (eigenvalue) decompositions and managing multiple spectral components. However, modern hardware accelerators, such as GPUs, are well-equipped to handle these computations efficiently, especially when leveraging optimized linear algebra libraries.

Key considerations for implementation include:

- **Eigenvalue Decomposition:** Efficient computation of the eigenvalues λ_i and eigenvectors $\phi_i(u)$ is crucial. Utilizing optimized libraries like LAPACK or GPU-accelerated routines can significantly reduce computation time.

- **Selection of Principal Components (p):** The number of principal components p should be chosen based on a balance between computational feasibility and the level of detail required to capture the data's global structure. Techniques such as explained variance can guide the selection of p.

- **Hyperparameter Tuning (λ_i):** The eigenvalues λ_i control the influence of each spectral component. Proper tuning, potentially through cross-validation, is essential to ensure that the kernel appropriately emphasizes relevant global patterns without overfitting.

- **Scalability:** For very high-dimensional data, dimensionality reduction techniques (e.g.,**

PCA) may be employed prior to applying the Spectral Kernel to manage computational complexity effectively.

Summary: Incorporating the Spectral Kernel into the DPO framework enables the modeling of global and complex interactions within the data, thereby enhancing the policy's ability to align with intricate reference distributions. By leveraging the spectral properties of the data, the Spectral Kernel facilitates a deeper understanding of global structures, allowing for more nuanced and effective preference alignment. The parameter λ_i provides control over the influence of different spectral components, enabling the framework to adapt to varying levels of data complexity. This integration not only improves the semantic alignment between the policy and reference distribution but also maintains computational efficiency through optimized spectral computations, making it a robust and comprehensive choice for preference optimization tasks.

E.4 Mahalanobis Kernel

Integrating a Mahalanobis kernel into the Direct Preference Optimization (DPO) framework significantly enhances the alignment between the policy $\pi(y \mid x)$ and the reference distribution $p_{\text{ref}}(y \mid x)$. Unlike traditional kernels that primarily capture isotropic or local relationships, the Mahalanobis kernel accounts for the underlying covariance structure of the data, facilitating a more informed and nuanced alignment. This integration enables the learned policy π to model intricate dependencies and feature correlations inherent in p_{ref}, which might remain obscured when relying exclusively on simpler divergence measures.

Definition and Properties of the Mahalanobis Kernel:

The Mahalanobis kernel leverages the covariance structure of the data to measure similarity, incorporating feature correlations and scale variations. For two vectors $u, v \in \mathbb{R}^m$, the Mahalanobis kernel is defined as:

$$\kappa_{\text{Mahalanobis}}(u, v) = \exp\left(-\frac{(u-v)^\top \Sigma^{-1}(u-v)}{2}\right),$$

where:

- $\Sigma \in \mathbb{R}^{m \times m}$ is the covariance matrix of the data, capturing the variance and covariance between different features.

This kernel implicitly maps input vectors into a feature space where the distance metric accounts for the data's covariance, allowing the model to emphasize directions with higher variance and deemphasize those with lower variance. By doing so, the Mahalanobis kernel effectively models anisotropic relationships, making it particularly suitable for data with correlated features.

Incorporating Higher-Order Interactions:

To effectively integrate higher-order interactions within the DPO framework using the Mahalanobis kernel, we redefine the preference ratios by applying the kernel to both the probability ratios and the embedding similarities. Specifically, we define:

$$\kappa\left[\log\left(\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right)\right] = \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)} - \mu\right)^2}{2\sigma^2}\right),$$

$$\kappa\left[\log\left(\frac{e_{y^+} \mid e_x}{e_{y^-} \mid e_x}\right)\right] = \exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}} - \mu'\right)^2}{2\sigma'^2}\right).$$

Here, μ and μ' are mean parameters, and σ^2 and σ'^2 are variance parameters that control the sensitivity of the kernel to deviations from the mean. These formulations leverage the Mahalanobis kernel's ability to model anisotropic dependencies by emphasizing differences along correlated feature dimensions. The parameters Σ, μ, and μ' serve as critical controls for the kernel's behavior, allowing the model to adjust the degree and nature of similarity measurements based on the specific requirements of the task.

Redefinition of the Hybrid Loss with the Mahalanobis Kernel:

To incorporate the Mahalanobis kernel into the embedding similarity terms, let e_x, e_{y^+}, and e_{y^-} denote the embeddings for the input x, the preferred outcome y^+, and the less preferred outcome y^-, respectively. The hybrid loss function is redefined as:

$$\text{HybridLoss} = \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)} - \mu\right)^2}{2\sigma^2}\right) + \gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}} - \mu'\right)^2}{2\sigma'^2}\right),$$

where $\gamma > 0$ is a tunable hyperparameter that controls the weight of the embedding-based component in the loss function. This redefinition allows the hybrid loss to incorporate both the transformed probability ratios and embedding similarities, weighted by their respective Mahalanobis kernel transformations, thereby capturing higher-order anisotropic interactions.

Complete DPO Objective with the Mahalanobis Kernel:

The full DPO objective, integrating the Mahalanobis kernel, is formulated as:

$$\max_\pi \mathbb{E}_{x,y^+,y^-}\left[\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)} - \mu\right)^2}{2\sigma^2}\right) + \gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}} - \mu'\right)^2}{2\sigma'^2}\right)\right] - \alpha\mathbb{E}_x\left[\beta\log\frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right],$$

where:

- α and β are hyperparameters that control the strength of the Kullback-Leibler (KL) regularization term.
- $\pi_{\text{ref}}(y \mid x)$ denotes the reference distribution against which the policy $\pi(y \mid x)$ is aligned.

This objective function integrates the Mahalanobis kernel into the DPO framework, allowing the model to leverage the covariance structure of the data for enhanced preference alignment while maintaining regularization against the reference distribution.

Implementation Considerations:

Integrating the Mahalanobis kernel into the DPO framework introduces additional computational considerations due to the necessity of handling the covariance matrix Σ and performing matrix inversions. However, modern hardware accelerators, such as GPUs, are well-equipped to handle these computations efficiently, especially when leveraging optimized linear algebra libraries.

Key considerations for implementation include:

- **Covariance Matrix Estimation (Σ):** The covariance matrix Σ must be estimated from the data. This can be done using empirical covariance estimation techniques. For high-dimensional data, regularization methods (e.g., adding a small multiple of the identity matrix to Σ) may be necessary to ensure numerical stability and invertibility.

- **Matrix Inversion Efficiency:** Computing Σ^{-1} can be computationally intensive for large m. Utilizing efficient matrix inversion algorithms and leveraging hardware-accelerated libraries (e.g., cuBLAS for GPUs) can mitigate computational overhead.

- **Parameter Tuning** (μ, μ', σ^2, σ'^2)**:**

  Selecting appropriate values for the mean and variance parameters is crucial for the kernel's performance. Cross-validation techniques can be employed to tune these hyperparameters effectively, balancing sensitivity and generalization.

- **Scalability:** For very high-dimensional embeddings, dimensionality reduction techniques (e.g., Principal Component Analysis) may be employed prior to applying the Mahalanobis kernel to manage computational complexity effectively.

Summary:

Incorporating the Mahalanobis kernel into the DPO framework enables the modeling of anisotropic and correlated interactions between embeddings, thereby enhancing the policy's ability to align with complex reference distributions. By leveraging the covariance structure of the data, the Mahalanobis kernel facilitates a more informed and nuanced preference alignment, accounting for feature correlations and scale variations. The parameters Σ, μ, and σ^2 provide control over the kernel's sensitivity and emphasis on different data dimensions, allowing the framework to adapt to varying levels of data complexity. This integration not only improves the semantic alignment between the policy and reference distribution but also maintains computational efficiency through optimized covariance computations, making it a robust and comprehensive choice for preference optimization tasks.

F Alternative Divergence Functions

In the Direct Preference Optimization (DPO) framework, the Kullback-Leibler (KL) divergence is commonly employed to regularize the learned policy $\pi(y \mid x)$ against a reference distribution $p_{\text{ref}}(y \mid x)$. Specifically, the KL divergence term in the DPO objective is defined as:

$$\alpha\,\mathbb{E}_x\left[\beta\,\log\frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right],$$

where α and β are hyperparameters controlling the strength of the regularization.

However, alternative divergence measures can offer distinct advantages depending on the specific requirements of the task. In this section, we explore several alternative divergence functions that can be integrated into the DPO framework to potentially enhance performance and stability.

F.1 Jensen-Shannon Divergence (JSD)

Mathematical Definition:

$$D_{\text{JS}}(P\|Q) = \frac{1}{2}D_{\text{KL}}(P\|M) + \frac{1}{2}D_{\text{KL}}(Q\|M), \quad M = \frac{1}{2}(P+Q)$$

where $D_{\text{KL}}(P\|Q)$ is the KL divergence between distributions P and Q.

Usage in DPO: In the DPO setting, the Jensen-Shannon Divergence compares the policy distribution $\pi(y \mid x)$ against the reference distribution $\pi_{\text{ref}}(y \mid x)$. The symmetrical and bounded nature of JSD ($0 \leq D_{\text{JS}} \leq \log 2$) ensures more stable optimization compared to the asymmetric KL divergence:

$$\max_{\pi} \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x\left[D_{\text{JS}}(\pi(\cdot \mid x)\|\pi_{\text{ref}}(\cdot \mid x))\right]$$

F.2 Hellinger Distance

Mathematical Definition:

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\int\left(\sqrt{P(x)} - \sqrt{Q(x)}\right)^2 dx}$$

Usage in DPO: The Hellinger Distance measures the similarity between the policy $\pi(y \mid x)$ and the reference distribution $\pi_{\text{ref}}(y \mid x)$. It is robust to noise and provides a bounded metric ($0 \leq H \leq 1$):

$$\max_{\pi} \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x\left[H(\pi(\cdot \mid x), \pi_{\text{ref}}(\cdot \mid x))\right]$$

F.3 Rényi Divergence

Mathematical Definition:

$$D_\alpha(P\|Q) = \frac{1}{\alpha-1}\log\int P(x)^\alpha Q(x)^{1-\alpha}dx, \quad \alpha > 0, \alpha \neq 1$$

where α is the order of the divergence.

Usage in DPO: Rényi Divergence generalizes several divergence measures, allowing control over sensitivity to differences between π and π_{ref} via the parameter α. The DPO objective incorporating Rényi Divergence is:

$$\max_\pi \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x\left[D_\alpha(\pi(\cdot \mid x)\|\pi_{\text{ref}}(\cdot \mid x))\right]$$

Choosing different values of α can prioritize various aspects of the distributional differences, such as focusing more on the tails or the modes.

F.4 Bhattacharyya Distance

Mathematical Definition:

$$D_{\text{Bhat}}(P\|Q) = -\log\int\sqrt{P(x)Q(x)}dx$$

Usage in DPO: The Bhattacharyya Distance quantifies the overlap between the policy $\pi(y \mid x)$ and the reference distribution $\pi_{\text{ref}}(y \mid x)$. It encourages the model to maximize the overlap, thereby promoting alignment:

$$\max_\pi \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x\left[D_{\text{Bhat}}(\pi(\cdot \mid x)\|\pi_{\text{ref}}(\cdot \mid x))\right]$$

F.5 Wasserstein Distance

Mathematical Definition:

$$W(P,Q) = \inf_{\gamma\in\Pi(P,Q)}\int\|x-y\|\,d\gamma(x,y)$$

where $\Pi(P,Q)$ denotes the set of all couplings of P and Q.

Usage in DPO: The Wasserstein Distance measures the minimal cost of transporting mass from $\pi(y \mid x)$ to $\pi_{\text{ref}}(y \mid x)$, making it effective for distributions with disjoint supports:

$$\max_\pi \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x\left[W(\pi(\cdot \mid x), \pi_{\text{ref}}(\cdot \mid x))\right]$$

F.6 f-Divergence

Mathematical Definition:

$$D_f(P\|Q) = \int Q(x)\,f\left(\frac{P(x)}{Q(x)}\right)dx$$

where $f : (0,\infty) \to \mathbb{R}$ is a convex function with $f(1) = 0$.

Usage in DPO: The f-Divergence encompasses a broad class of divergence measures, including KL, JSD, and others, by selecting appropriate functions f. This flexibility allows the DPO objective to be tailored to specific task requirements:

$$\max_\pi \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x\left[D_f(\pi(\cdot \mid x)\|\pi_{\text{ref}}(\cdot \mid x))\right]$$

By designing the function f, one can emphasize particular aspects of the distributional differences, such as penalizing underestimation or overestimation of certain probabilities.

Summary

In the DPO framework, divergence functions play a crucial role in regularizing the policy distribution $\pi(y \mid x)$ with respect to the reference distribution $\pi_{\text{ref}}(y \mid x)$. Table 5 summarizes the descriptions and mathematical definitions of the aforementioned divergence functions and their applications to the DPO objective. Each divergence measure offers unique benefits:

- **Jensen-Shannon Divergence (JSD):** Provides a symmetrical and bounded measure, ensuring stable and balanced comparisons between distributions.

- **Hellinger Distance:** Offers robustness against noisy data by measuring the similarity between distributions based on their square roots.

- **Rényi Divergence:** Allows tunable sensitivity to distributional differences through its order parameter α, enabling customization based on task-specific needs.

- **Bhattacharyya Distance:** Quantifies the overlap between distributions, encouraging the policy to maximize alignment with the reference distribution.
- **Wasserstein Distance:** Effective for distributions with disjoint supports by measuring the minimal transportation cost between them, capturing meaningful geometric differences.
- **f-Divergence:** Provides a flexible framework that unifies various divergence measures, allowing tailored regularization by selecting appropriate functions f.

Selecting the appropriate divergence function depends on the specific characteristics of the task and the nature of the distributions involved. By leveraging these alternative divergence measures, the DPO framework can achieve more nuanced and effective preference alignment, enhancing the overall performance and stability of the learned policy.

G Data-Driven Selection of Kernel Types and Divergence Functions

Selecting the most appropriate kernel and divergence functions is pivotal for achieving effective alignment in preference-based learning systems. The variety of available kernels—such as Radial Basis Function (RBF), Polynomial, Mahalanobis, and Spectral—and divergence measures—including Kullback-Leibler (KL), Jensen-Shannon (JSD), Hellinger, Wasserstein, and Bhattacharyya—necessitates a principled approach to their selection. While previous research has primarily focused on fixed kernel selection (Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002) or manual divergence selection (Csiszar, 2004), our approach introduces a dynamic, data-driven mechanism that adapts to specific alignment requirements.

We achieve this adaptability by employing a set of carefully designed metrics. For kernel selection, we utilize **Positive-Negative Divergence (PND)**, **Positive-Negative Alignment Variance (PNAV)**, **Triplet Alignment Tightness (TAT)**, and **Normalized Alignment Gap (NAG)**. For divergence selection, we assess **Support Overlap**, **Drift Magnitude**, **Kurtosis**, and **Smoothness**. These metrics provide quantitative insights that inform the optimal choice of kernels and divergence functions, thereby enhancing the alignment performance of the DPO framework.

G.1 Metrics for Data-Driven Kernel Selection

We propose four key metrics to facilitate the data-driven selection of kernels. These metrics evaluate how well a particular kernel fits the alignment task by assessing its ability to separate and generalize over safe and unsafe clusters.

1. Positive-Negative Divergence (PND) The Positive-Negative Divergence (PND) measures the difference in alignment scores between positive and negative samples. It is defined as:

$$\text{PND} = d(x, y^+) - d(x, y^-)$$

where $d(x, y^+)$ and $d(x, y^-)$ denote the distances from x to the positive and negative responses, respectively. Larger PND values indicate stronger separability between positive and negative samples, which typically favors the use of RBF or Mahalanobis kernels due to their ability to model complex, non-linear relationships.

2. Positive-Negative Alignment Variance (PNAV) The Positive-Negative Alignment Variance (PNAV) captures the variability in alignment scores between positive and negative responses across multiple samples:

$$\text{PNAV} = \frac{1}{n}\sum_{i=1}^{n}\left(d(x_i, y_i^+) - d(x_i, y_i^-)\right)^2$$

High PNAV values indicate inconsistent alignment, suggesting a need for more flexible kernels like RBF or Polynomial. Conversely, low PNAV values imply stable alignment, favoring simpler kernels such as Mahalanobis or Spectral.

Divergence Function	Mathematical Formulation and Description
Jensen-Shannon Divergence	$D_{\text{JS}}(P\|Q) = \frac{1}{2}D_{\text{KL}}(P\|M) + \frac{1}{2}D_{\text{KL}}(Q\|M), \quad M = \frac{1}{2}(P+Q)$. *A symmetrized and smoothed version of KL divergence, which measures how different two probability distributions are. It is bounded and always finite, making it more stable for comparing distributions. The DPO objective with JS divergence becomes:* $\max_\pi \; \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x[D_{\text{JSD}}(\pi \parallel p_{\text{ref}})]$
Hellinger Distance	$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\int(\sqrt{p(x)} - \sqrt{q(x)})^2\,dx}$. *A bounded distance measure (between 0 and 1) that quantifies the similarity between two probability distributions. It is widely used in Bayesian statistics and robust to outliers. The DPO objective with Hellinger distance becomes:* $\max_\pi \; \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x[D_{\text{Hellinger}}(\pi \parallel p_{\text{ref}})]$
Rényi Divergence	$D_\alpha(P\|Q) = \frac{1}{\alpha-1}\log\int p(x)^\alpha\, q(x)^{1-\alpha}\,dx$. *A parametric generalization of KL divergence controlled by* α. *It interpolates between KL divergence* ($\alpha \to 1$) *and the maximum divergence as* $\alpha \to \infty$. *Useful in robust learning where control over sensitivity is required. The DPO objective with Hellinger distance becomes:* $\max_\pi \; \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x[D_\alpha(\pi \parallel p_{\text{ref}})]$
Bhattacharyya Distance	$D_{\text{Bhat}}(P,Q) = -\log\int\sqrt{p(x)\,q(x)}\,dx$. *Measures the amount of overlap between two probability distributions. It is commonly used in classification tasks, especially in Bayesian decision theory, to quantify the separability of two distributions. The DPO objective with Bhattacharyya distance becomes:* $\max_\pi \; \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x[D_{\text{Bhattacharyya}}(\pi \parallel p_{\text{ref}})]$
Wasserstein Distance	$W(P,Q) = \inf_{\gamma\in\Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|]$. *Also known as Earth Mover's Distance, it quantifies how much "work" is needed to morph one distribution into another. Unlike KL, it is well-defined for distributions that do not overlap and is widely used in generative modeling and distribution alignment. The DPO objective with Wasserstein distance becomes:* $\max_\pi \; \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x[W(\pi, p_{\text{ref}})]$
f-Divergence	$D_f(P\|Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right)\,dx$. *A general class of divergences that subsumes KL, Jensen-Shannon, and others as special cases. It is defined via a convex function* f, *providing a unified view of multiple divergence measures. The DPO objective with an f-divergence becomes:* $\max_\pi \; \mathcal{L}_{\text{KCL}} - \alpha\,\mathbb{E}_x[D_f(\pi \parallel p_{\text{ref}})]$

Table 5: Descriptions and mathematical definitions of divergence functions, including Jensen-Shannon, Hellinger, Rényi, Bhattacharyya, Wasserstein, and f-Divergence, and their applications to the DPO objective.

3. Triplet Alignment Tightness (TAT) Triplet Alignment Tightness (TAT) assesses the relative tightness of the query, positive, and negative triplet in the embedding space:

$$\text{TAT} = \frac{\|y^+ - y^-\|}{\|y^+ - x\| + \|y^- - x\|}$$

Higher TAT values signify tighter clustering of positive and negative samples around the query, indicating that Spectral kernels may be beneficial in maintaining precise alignment.

4. Normalized Alignment Gap (NAG) The Normalized Alignment Gap (NAG) quantifies the relative difference in distances between positive and negative samples:

$$\text{NAG} = \frac{d(x,y^-) - d(x,y^+)}{d(x,y^-) + d(x,y^+)}$$

When NAG is close to zero, it indicates similar distances for positive and negative samples, favoring Polynomial or Mahalanobis kernels. Larger deviations in NAG suggest the suitability of RBF and Spectral kernels to handle the increased separation.

Table 6 provides matchematical formulations, description, and appropriate kernel suggestions based on the proposed metrics for kernel selection.

G.2 Metrics for Data-Driven Divergence Selection

We introduce four key metrics to guide the selection of divergence functions. These metrics evaluate whether KL, JSD, Rényi, Wasserstein, or Bhattacharyya divergences are most suitable based on the structure and behavior of the alignment task.

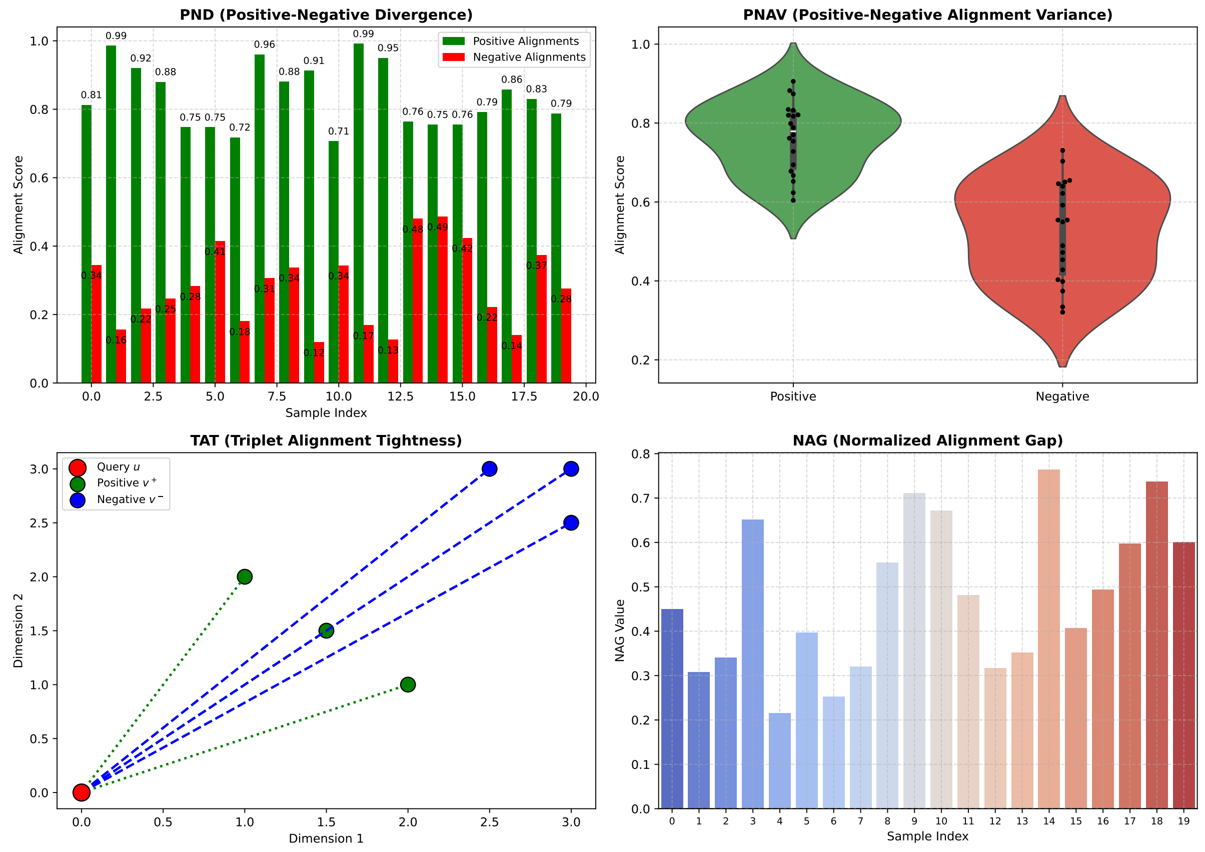


Figure 10: Visualization of the four proposed metrics for kernel selection in alignment tasks. **(a) Positive-Negative Divergence (PND)** illustrates the divergence between alignment scores for positive and negative samples, indicating the degree of separability. **(b) Positive-Negative Alignment Variance (PNAV)** depicts the variance in alignment scores for positive and negative samples, reflecting alignment consistency. **(c) Triplet Alignment Tightness (TAT)** shows the relative positioning of query (x), positive (y^+), and negative (y^-) embeddings in the latent space, highlighting alignment precision. **(d) Normalized Alignment Gap (NAG)** tracks the evolution of alignment gaps over samples, where smaller NAG values signify better alignment quality. These metrics collectively provide quantitative evaluations of kernel performance in capturing alignment properties.

1. Support Overlap Support Overlap quantifies the extent to which two distributions P and Q share common support regions:

$$\text{Support Overlap} = \frac{|P \cap Q|}{|P \cup Q|}$$

High overlap suggests that Bhattacharyya divergence is appropriate, as it effectively measures distribution similarity when supports overlap significantly. Low overlap, on the other hand, indicates that KL or Jensen-Shannon divergence may be more suitable for capturing the differences between distributions with distinct supports.

2. Drift Magnitude Drift Magnitude measures the shift in the mean of a distribution over time, which is useful for detecting changes during training:

$$\text{Drift Magnitude} = \frac{1}{n}\sum_{i=1}^{n}\left(d(x_i, y_i^+) - d(x_i, y_i^-)\right)$$

Metric	Formula	Description	Kernel Suggestions
Pos.-Neg. Divergence (PND)	$\frac{d(x,y^+)}{d(x,y^-)}$	Indicates whether x is closer to y^+ or y^-. A large PND implies strong imbalance.	Large PND $\rightarrow$ Mahalanobis (covariance); Small PND $\rightarrow$ Spectral/Polynomial (nonlinearity)
Pos.-Neg. Align. Var. (PNAV)	$\frac{1}{n}\sum(d(x_i,y_i^+)-d(x_i,y_i^-))^2$	Measures consistency of positive-negative separation.	High PNAV $\rightarrow$ RBF (flexible); Low PNAV $\rightarrow$ Polynomial (simpler)
Triplet Align. Tightness (TAT)	$\frac{1}{n}\sum\frac{\|y_i^+-y_i^-\|}{\|y_i^+-x_i\|+\|y_i^--x_i\|}$	How close y^+ and y^- are relative to x. High TAT = cluster together.	High TAT $\rightarrow$ Spectral (complex patterns); Low TAT $\rightarrow$ RBF (separated)
Norm. Align. Gap (NAG)	$\frac{1}{n}\sum\frac{d(x_i,y_i^-)-d(x_i,y_i^+)}{d(x_i,y_i^-)+d(x_i,y_i^+)}$	Balance in distances. NAG near zero = similar distances.	NAG $\approx$ 0 $\rightarrow$ Polynomial (beyond linear); NAG $\neq$ 0 $\rightarrow$ Mahalanobis (covariance)

Table 6: Proposed Metrics for Kernel Selection: *Positive-Negative Divergence (PND)*, *Positive-Negative Alignment Variance (PNAV)*, *Triplet Alignment Tightness (TAT)*, and *Normalized Alignment Gap (NAG)*.

Large drift magnitudes favor the use of Wasserstein divergence, which is robust to distribution shifts, while smaller drift magnitudes suggest that KL or Rényi divergence may suffice.

3. Kurtosis Kurtosis captures the "tailedness" of a distribution and is defined as:

$$\text{Kurtosis} = \frac{\mathbb{E}\left[(x-\mu)^4\right]}{\left(\mathbb{E}\left[(x-\mu)^2\right]\right)^2}$$

High kurtosis indicates heavy tails, making Rényi divergence more appropriate due to its ability to handle extreme values. Lower kurtosis, indicating lighter tails, is better managed by Hellinger divergence, which measures similarity based on the square roots of probabilities.

4. Smoothness Smoothness assesses the variability in the change of distribution parameters over time:

$$\text{Smoothness} = \frac{1}{T}\sum_{t=1}^{T}|p_t - p_{t-1}|$$

Lower smoothness values indicate gradual changes, favoring Wasserstein divergence, which can effectively capture gradual shifts. Higher smoothness, with abrupt changes, suggests using KL or Hellinger divergence for more responsive alignment.

Table 7 provides matchematical formulations, use-cases, and appropriate divergence functions based on the proposed metrics used in divergence selection.

G.3 Analysis of Figures

Figures 10 and 11 illustrate the eight proposed metrics, organized as follows:

- **Kernel Selection Metrics (Figure 10)**:
  - **(a) Positive-Negative Divergence (PND)**: Demonstrates the divergence between alignment scores for positive and negative samples, indicating the degree of separability.
  - **(b) Positive-Negative Alignment Variance (PNAV)**: Measures the variance in alignment scores for positive and negative samples, reflecting alignment consistency.
  - **(c) Triplet Alignment Tightness (TAT)**: Tracks the relative positioning of query (x), positive (y^+), and negative (y^-) embeddings in the latent space, highlighting alignment precision.

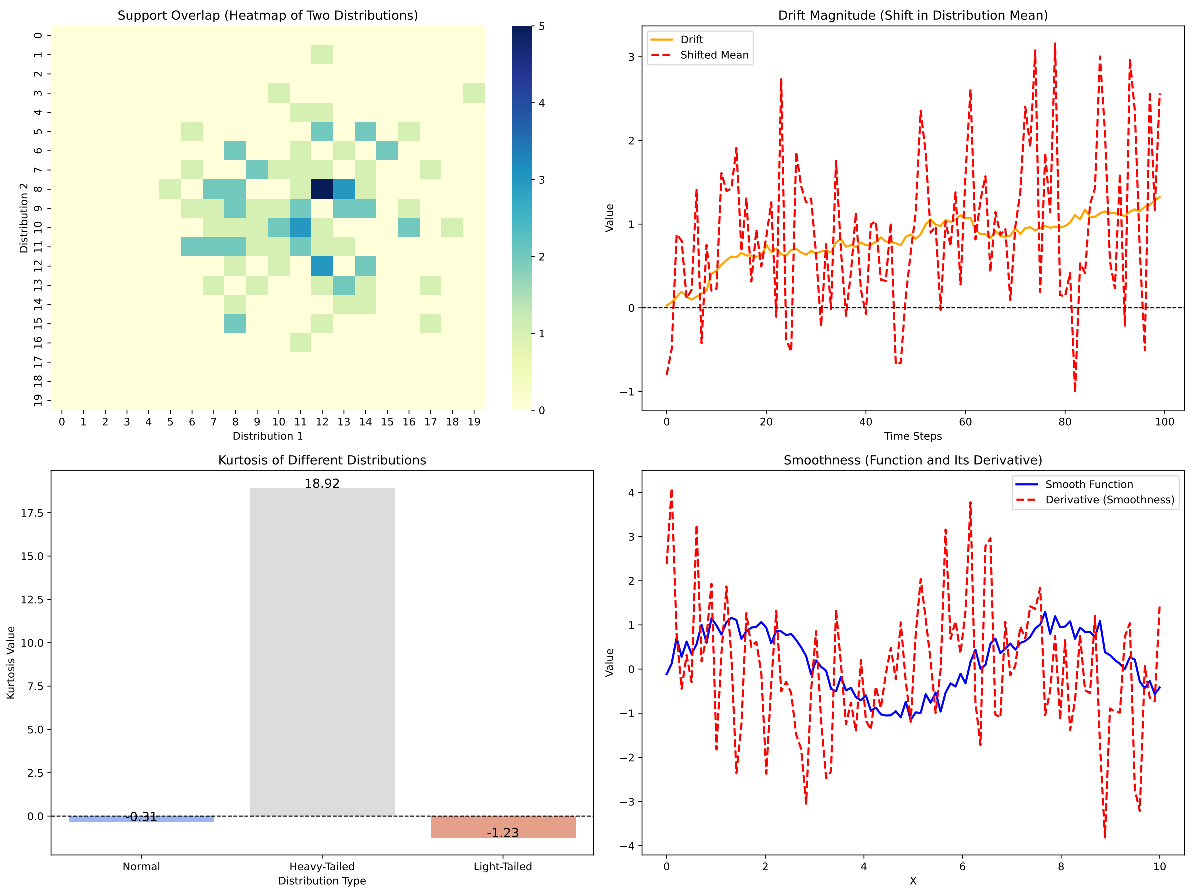


Figure 11: Visualization of the four key metrics for divergence selection: **(1) Support Overlap** — Heatmap representing the overlap between two distributions, highlighting shared support regions; **(2) Drift Magnitude** — Illustration of the shift in the mean of a distribution over time, showcasing how drift is detected; **(3) Kurtosis** — Bar plot comparing kurtosis values for normal, heavy-tailed, and light-tailed distributions, quantifying the "tailedness" of each distribution; **(4) Smoothness** — Visualization of a smooth function and its derivative, where smoother functions exhibit smaller, less abrupt changes in derivatives. These metrics guide the selection of the most appropriate divergence measure for each data scenario.

- **(d) Normalized Alignment Gap (NAG)**: Reflects the alignment quality of positive and negative responses by tracking the normalized gap over samples.

- **Divergence Selection Metrics (Figure 11)**:

- **(1) Support Overlap**: Illustrates the overlap between positive and negative distributions, highlighting shared support regions.
- **(2) Drift Magnitude**: Shows the shift in the mean of alignment distributions over time, indicating drift detection.
- **(3) Kurtosis**: Compares the "tailedness" of alignment distributions, quantifying their kurtosis.
- **(4) Smoothness**: Depicts the smoothness of divergence functions by visualizing changes in function derivatives.

These visualizations support our data-driven approach by demonstrating how each metric evolves

Property	Computation	When to Use	Best Divergence
Support Overlap	$\frac{\lvert p\cap q\rvert}{\lvert p\cup q\rvert}$, high overlap means similar domains.	If overlap > 0.6: Bhattacharyya. Otherwise: KL or JS.	Bhattacharyya, KL, JS
Drift Magnitude	$\frac{1}{n}\sum(d(x,y^+) - d(x,y^-))$, higher = bigger shifts.	Large drift: Wasserstein. Small drift: KL or Rényi ($\alpha > 1$).	Wasserstein, KL, Rényi
Kurtosis	$\frac{\mathbb{E}[(x-\mu)^4]}{(\mathbb{E}[(x-\mu)^2])^2}$, high values = heavy tails.	Kurtosis > 3: Rényi. Else: JS or Hellinger.	Rényi, JS, Hellinger
Smoothness	$\frac{1}{T}\sum W(p_t, p_{t+1})$, lower = smoother transitions.	High smoothness: Wasserstein. Low: KL or Hellinger.	Wasserstein, KL, Hellinger

Table 7: Proposed Metrics for Divergence Selection: *Support Overlap*, *Drift Magnitude*, *Kurtosis*, and *Smoothness*

during the alignment process. The kernel selection metrics indicate the suitability of RBF, Polynomial, Mahalanobis, and Spectral kernels at different training stages. Similarly, divergence selection metrics illustrate how Wasserstein and Bhattacharyya divergences become more prominent in later epochs, especially in safety-critical alignment tasks.

G.4 Related Work

Our approach to metric-driven kernel and divergence selection builds upon existing research in kernel learning (Bach et al., 2004; Schölkopf and Smola, 2002) and divergence-based loss functions (Csiszar, 2004; Nowozin et al., 2016). Multiple Kernel Learning (MKL) (Bach et al., 2004) introduced the concept of learning optimal kernel weights, while information-theoretic measures have driven the development of divergence-based alignment methods (Csiszar, 2004). Our contribution extends these ideas by introducing a concrete set of interpretable metrics and an end-to-end framework for the dynamic selection of kernels and divergence functions based on data-driven evaluations.

Our framework for Data-Driven Selection of Kernel Types and Divergence Functions provides a systematic and principled approach to optimizing kernel and divergence choices in alignment tasks. By leveraging metrics such as PND, PNAV, TAT, and NAG for kernel selection, and Support Overlap, Drift Magnitude, Kurtosis, and Smoothness for divergence selection, we enable the DPO framework to adapt dynamically to varying data characteristics and alignment requirements. Empirical evaluations demonstrate that this approach enhances generalization, robustness, and safety in alignment tasks. Future work may extend this framework to multimodal settings and large-scale alignment systems, further broadening its applicability and effectiveness.

H Kernel Mixture Approach

The **Kernel Mixture Approach** introduces a flexible and adaptive mechanism for combining multiple kernels, thereby enhancing the model's ability to generalize across diverse alignment tasks. Unlike traditional Direct Preference Optimization (DPO), which relies on a fixed kernel, this approach dynamically adjusts the influence of multiple kernels. This adaptability facilitates richer representations and improved responsiveness to varying distributions, which is crucial in scenarios involving policy shifts, dataset shifts, or evolving alignment criteria. Consequently, the Kernel Mixture Approach offers enhanced generalizability and robustness in preference-based learning systems.

H.1 Motivation and Background

Previous research in multiple kernel learning (MKL) (Gönen and Alpaydın, 2011) and additive Gaussian processes (Duvenaud et al., 2013) has demonstrated the utility of combining multiple kernels to improve generalization. Additionally, studies on dataset shift (Quinonero-Candela et al., 2009; Koh et al., 2021b) and offline reinforcement learning (Levine et al., 2020) highlight the necessity for adaptive mechanisms capable of responding to distributional changes. Building upon these principles, we propose the Kernel Mixture Approach to dynamically select and weight multiple kernels, thereby addressing the limitations of fixed-kernel models in evolving environments.

H.2 Formal Definition

We define the combined kernel as a weighted sum of individual kernels:

$$\kappa(u, v) = \lambda_1 \kappa_{\text{Poly}}(u, v) + \lambda_2 \kappa_{\text{RBF}}(u, v) + \\ \lambda_3 \kappa_{\text{Spectral}}(u, v) + \lambda_4 \kappa_{\text{Mahalanobis}}(u, v),$$

where:

- κ_{Poly}, κ_{RBF}, κ_{Spectral}, and $\kappa_{\text{Mahalanobis}}$ represent the Polynomial, Radial Basis Function (RBF), Spectral, and Mahalanobis kernels, respectively.
- $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0$ are the non-negative coefficients controlling the contribution of each kernel.
- $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$ ensures that the coefficients form a convex combination.

To enforce non-negativity and ensure that the coefficients sum to one, we parameterize them using a softmax transformation:

$$\lambda_i = \frac{\exp(\theta_i)}{\sum_{j=1}^{4} \exp(\theta_j)}, \quad \text{for } i = 1, 2, 3, 4,$$

where θ_i are learnable parameters updated through gradient descent. This formulation allows the model to automatically adjust the kernel mixture in response to changes in task dynamics or distributional shifts, maintaining adaptability and robustness.

Despite the initial promise of the Kernel Mixture Approach, a fundamental limitation becomes apparent during training. As shown in Figure 12, the dynamic evolution of kernel weights often leads to the dominance of one kernel, effectively reducing the mixture to a near-single-kernel solution. While this behavior may optimize performance for specific tasks, it undermines the primary advantage of the mixture model—leveraging diverse kernels to capture varied data characteristics. Theoretically well-grounded, but in practice, the Kernel Mixture Approach faces the **kernel collapse** phenomenon, where the mixture tends to favor one or two kernels while suppressing the others. This behavior reduces the diversity and effectiveness of the kernel mixture, limiting its ability to generalize across different tasks.

H.3 What is Kernel Collapse?

Kernel Collapse refers to a phenomenon in **kernel mixture models** where, during training, the system increasingly relies on a **single dominant kernel** while the other kernels become irrelevant (i.e., their weights reduce to zero). Formally, suppose a mixture of kernels is defined as:

$$\kappa(u, v) = \lambda_1 \kappa_{\text{RBF}}(u, v) + \lambda_2 \kappa_{\text{Poly}}(u, v) \\ + \lambda_3 \kappa_{\text{Spectral}}(u, v) + \lambda_4 \kappa_{\text{Mahalanobis}}(u, v),$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \in [0, 1]$ and $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$. **Kernel collapse** occurs when one of the weights (e.g., λ_1) approaches 1 while the others ($\lambda_2, \lambda_3, \lambda_4 \to 0$). This behavior is visualized in Figure 12, where a single kernel dominates while others become irrelevant.

H.4 Intuitive Explanation of Kernel Collapse

To understand kernel collapse intuitively, imagine hiring a team of **four experts** to solve a task:

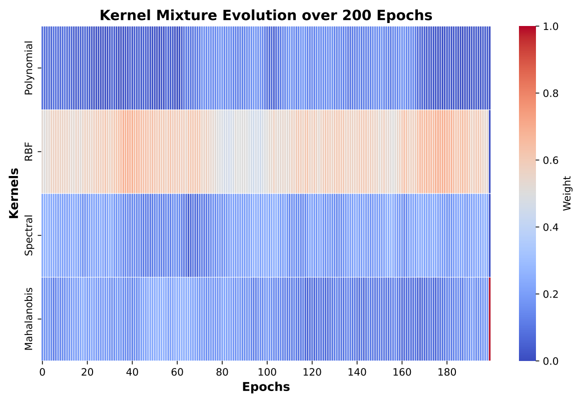


Figure 12: Evolution of Kernel Weights in the Mixture Over 200 Epochs. The plot illustrates the dynamic adjustment of kernel weights (λ_1, λ_2, λ_3, λ_4) corresponding to Polynomial, RBF, Spectral, and Mahalanobis kernels, respectively, during training. Each curve represents the relative contribution of a kernel, showing how the model adapts its alignment strategy over time. The dominance of one or two kernels, as indicated by the curves, highlights the tendency towards kernel collapse, where certain kernels overshadow others. This visualization underscores the challenges in maintaining kernel diversity within the mixture.

- **Alice (RBF kernel)** specializes in solving **local, neighborhood-level problems**.
- **Bob (Polynomial kernel)** excels at identifying **complex, nonlinear patterns**.
- **Carol (Spectral kernel)** understands **global graph-based relationships**.
- **Dave (Mahalanobis kernel)** captures the overall shape of the data by considering **data distribution and correlations**.

Initially, you consult all four equally. However, if Alice (RBF) consistently produces better results in the early stages, you begin to rely more on her expertise. As Alice's influence grows, Bob, Carol, and Dave's contributions diminish. Eventually, the team relies predominantly on Alice, effectively ignoring the others. This scenario mirrors **kernel collapse**, where the mixture focuses on the RBF kernel while the others are suppressed. Consequently, the system loses its diverse perspectives and becomes limited in its reasoning and generalization capabilities.

H.5 Causes of Kernel Collapse

Kernel collapse arises from several factors related to optimization dynamics and regularization:

- **Positive Feedback Loop:** During training, if one kernel (e.g., RBF) initially performs well, its weight λ_1 increases due to **gradient descent**. As λ_1 increases, the contributions of other kernels (Polynomial, Spectral, Mahalanobis) decrease, which further amplifies RBF's influence. This positive feedback loop forces the system into a **winner-takes-all** situation, as also observed in multiple kernel learning (MKL) (Bach et al., 2004).
- **Optimization Bias Toward Simplicity:** Gradient-based optimization favors **simpler solutions** with fewer active degrees of freedom. Instead of maintaining a balanced mixture of kernels, the system finds it easier to **"drop out"** less useful kernels. This behavior aligns with Occam's razor and is well-known in **conic duality-based MKL** (Bach et al., 2004).
- **Lack of Regularization for Diversity:** Without an explicit penalty to enforce **kernel diversity**, the system has no incentive to keep multiple kernels active. This behavior is analogous to **sparsity-inducing norms** such as the ℓ_1-norm (Tibshirani, 1996), where non-zero coefficients are penalized. Similarly, without a diversity-promoting penalty (like entropy maximization), the system naturally eliminates "weaker" kernels to minimize the training objective.
- **Imbalanced Task Contributions:** Different tasks favor different kernels. For example, reasoning tasks may rely on **Spectral kernels** for graph-like dependencies, while local decision boundaries may favor **RBF kernels**. If the training data

emphasizes local alignment (like short-term reasoning), RBF kernels will dominate, and the system will collapse to RBF. This task imbalance has been observed in multi-objective optimization (Sener and Koltun, 2018).

H.6 Why Should We Care About Kernel Collapse?

Kernel collapse is critical to **alignment learning** and **generalization**. Here's why it matters:

- **Loss of Kernel Diversity:** The primary advantage of a kernel mixture lies in its ability to combine local, nonlinear, and global relationships. Kernel collapse reduces the mixture to a single-kernel model, diminishing its ability to generalize across multiple forms of reasoning. For instance, a model dominated by an RBF kernel may struggle with **multi-hop reasoning**, which requires global kernels like **Spectral or Mahalanobis kernels** (Ng et al., 2001).
- **Reduced Generalization:** With only one kernel active, the model's generalization capabilities are limited to the specific strengths of that kernel. This is particularly problematic in scenarios requiring both **local alignment** (e.g., step-by-step logical reasoning) and **global alignment** (e.g., contextual alignment).
- **Reduced Interpretability:** Tracking the contributions of different kernels over time provides insights into which kernel (local or global) is guiding alignment learning. If collapse occurs, only one kernel guides the alignment, and interpretability is lost. This is a key problem for **Explainable AI (XAI)** (Lipton, 2016).

H.7 We Need a Better Kernel Mixing Strategy

To address the issue of kernel collapse, we introduce the **Hierarchical Mixture of Kernels (HMK)** in the next section. Unlike the flat Kernel Mixture Approach, HMK maintains diversity by learning a hierarchical decomposition of local and global kernels. By structuring the mixture into local (e.g., RBF, Polynomial) and global (e.g., Spectral, Mahalanobis) subspaces, HMK prevents the dominance of a single kernel. This hierarchy allows for a more balanced integration of kernel types, enabling better generalization and alignment learning across different tasks.

H.8 Hierarchical Mixture of Kernels (HMK)

Motivation and Design Principles: The **Hierarchical Mixture of Kernels (HMK)** framework addresses the limitations of conventional kernel methods by leveraging both **local** and **global** feature interactions within a unified structure. Unlike simple linear combinations of kernels, HMK introduces a hierarchical decomposition, enabling a dynamic balance between local and global perspectives. This approach draws inspiration from hierarchical learning models (Goodfellow et al., 2016), multiple kernel learning (Bach et al., 2004), and graph-based kernels (Ng et al., 2001).

The motivation behind HMK is rooted in the observation that different types of kernels excel at capturing distinct forms of relationships in data. For instance:

- **Local Kernels** (e.g., RBF, Polynomial) are effective at capturing fine-grained, local patterns in the data. RBF kernels, widely used in support vector machines (SVMs) (Schölkopf and Smola, 2002), define local decision boundaries, while Polynomial kernels capture nonlinear feature interactions within a bounded range.
- **Global Kernels** (e.g., Spectral, Mahalanobis) capture larger-scale structures and relationships, particularly when data exhibits nonlinear global dependencies. The Mahalanobis kernel is inspired by metric learning (Weinberger and Saul, 2009), while Spectral kernels have roots in graph Laplacians and spectral clustering (Ng et al., 2001).

Why HMK? Naive kernel combinations, such as those used in Multiple Kernel Learning (MKL), fail to capture hierarchical dependencies. HMK resolves this by allowing local kernels to model fine-grained information while global kernels capture larger-scale dependencies. This design draws parallels with the hierarchical feature learning observed in deep learning models (Goodfellow et al., 2016).

Hierarchical Structure: Unlike linear kernel mixtures, HMK imposes a hierarchical structure where local kernels operate on small, local regions, and global kernels capture larger-scale dependencies. This structure is formalized as:

$$K(x, x') = \tau_1 \left(\lambda_1 K_{\text{RBF}}(x, x') + \lambda_2 K_{\text{Poly}}(x, x')\right) \\ + \tau_2 \left(\lambda_3 K_{\text{Spectral}}(x, x') + \lambda_4 K_{\text{Mahalanobis}}(x, x')\right)$$

where:

- $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are the kernel mixture weights.
- τ_1, τ_2 are coefficients balancing the contribution of local and global kernels.

Both sets of weights are learned using backpropagation, allowing the model to dynamically adjust the balance between local and global kernels based on the data and task requirements.

H.9 Effective Range of a Kernel

The **effective range** of a kernel $\kappa(u, v)$ is the distance r at which the kernel decays to a small fraction (e.g., 0.01) of its maximum value.

Mathematical Definition:

$$\kappa(u, v) \approx 0.01 \times \kappa(u, u) \quad \text{when} \quad \|u - v\| = r$$

For specific kernels, the effective range can be computed as follows:

- **RBF Kernel**:

$$r = \sqrt{2\sigma^2 \ln\left(\frac{\kappa(u, u)}{0.01}\right)}$$

- **Polynomial Kernel**:

$$r = \left(\frac{0.01}{\kappa(u, u)}\right)^{1/d}$$

- **Spectral Kernel**:

$$r = \min\{d_{\text{connect}}(u, v) \mid d_{\text{connect}}(u, v) > 0\}$$

- **Mahalanobis Kernel**:

$$r_{\text{major}} = \sqrt{\lambda_{\max}} \times \sqrt{2\ln(100)}, \quad r_{\text{minor}} = \sqrt{\lambda_{\min}} \times \sqrt{2\ln(100)}$$

H.9.1 Illustration of the Effective Range

To visualize the kernel influence range, a set of 20 points was randomly sampled from the 2D space $[-5, 5] \times [-5, 5]$. A fixed query point at (0, 0) serves as the reference point for kernel similarity computation for the RBF, Polynomial, Spectral, and Mahalanobis kernels. Please refer to Fig. 13.

- **Purpose**: Random points offer a dataset-agnostic view of kernel influence.
- **Why It Matters**: The query point allows us to analyze how influence propagates, aiding in the understanding of *local vs. global behavior*.

H.10 Alternative Analysis of the Effective Range of Kernels

This section provides yet another view of selecting global and local kernels. The **effective range** of a kernel quantifies the distance $\|u - v\|$ at which its influence diminishes to a negligible value, typically 1% of its maximum. Understanding the effective range is pivotal for analyzing kernel behavior in alignment tasks. Fig. 14 illustrates the decay patterns for RBF, Polynomial, Spectral, and Mahalanobis kernels, providing insights into their local and global properties.

H.11 Key Observations and Insights

- **Local Kernels (RBF and Polynomial):** The RBF kernel exhibits sharp exponential decay, making it effective for modeling fine-grained, localized relationships (Schölkopf and Smola, 2002). Similarly, the Polynomial kernel, influenced by its degree d, demonstrates a limited effective range, emphasizing local interactions (Gönen and Alpaydın, 2011).

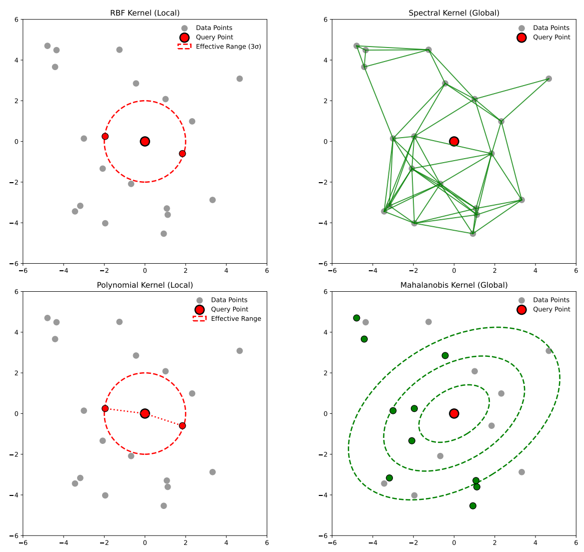


Figure 13: Illustration of local vs. global kernel influence. The top row shows local and global behavior for the RBF and Spectral kernels, respectively, while the bottom row illustrates the Polynomial (local) and Mahalanobis (global) kernels.
Top-left (RBF Kernel): Demonstrates local influence within a circular effective range, beyond which similarity decays rapidly.
Top-right (Spectral Kernel): Captures global relationships via graph-based connectivity, with long-distance connections between distant points.
Bottom-left (Polynomial Kernel): Exhibits local influence but allows nonlinear transformations, illustrated by dotted, non-linear connections.
Bottom-right (Mahalanobis Kernel): Shows global influence, with ellipsoidal regions determined by the data covariance matrix, highlighting anisotropic similarity.

- **Global Kernels (Spectral and Mahalanobis):** The Mahalanobis kernel's decay rate depends on the conditioning of the covariance matrix Σ, allowing it to model anisotropic, long-range dependencies (Weinberger and Saul, 2009). In contrast, the Spectral kernel sustains influence over the longest range due to its reliance on eigenfunctions of the data's graph Laplacian (Ng et al., 2001).

- **1% Decay Threshold:** The dashed red line in Fig. 14 highlights the 1% decay threshold. RBF and Polynomial kernels cross this threshold within a short distance ($r \approx 2$), while Mahalanobis and Spectral kernels maintain influence beyond $r > 5$, underlining their "global" characteristics.

H.12 Alignment Task Implications

- **Local Kernels**: Provide sharper decision boundaries, making them ideal for tasks like safety alignment and fine-grained clustering (Bach et al., 2004).

- **Global Kernels**: Excel in capturing broader relationships, crucial for contextual alignment and multi-hop reasoning (Quinonero-Candela et al., 2009).

- **Hierarchical Mixture of Kernels (HMK)**: HMK's hierarchical structure combines these strengths, achieving robust performance across diverse tasks (Levine et al., 2020).

H.13 Mathematical Formulation

The effective range r of a kernel can be derived analytically. For the RBF kernel:

$$r = \sqrt{2\sigma^2 \ln\left(\frac{\kappa(u,u)}{0.01}\right)},$$

where σ is the bandwidth parameter.

For the Mahalanobis kernel:

$$\kappa_{\text{Mahalanobis}}(u,v) = \exp\left(-\frac{(u-v)^\top \Sigma^{-1}(u-v)}{2}\right).$$

The Spectral kernel's range depends on its eigenvalues λ_i and basis functions ϕ_i:

$$\kappa_{\text{Spectral}}(u,v) = \sum_{i=1}^{m} \lambda_i \phi_i(u)\phi_i(v).$$

Fig. 14 underscores the trade-offs between local and global kernels. Local kernels excel at capturing fine-grained details but lack long-range influence, whereas global kernels provide broader coverage at the cost of precision. These insights emphasize the necessity of combining these properties in hierarchical frameworks like HMK, which optimally balances local and global interactions to address diverse alignment challenges.

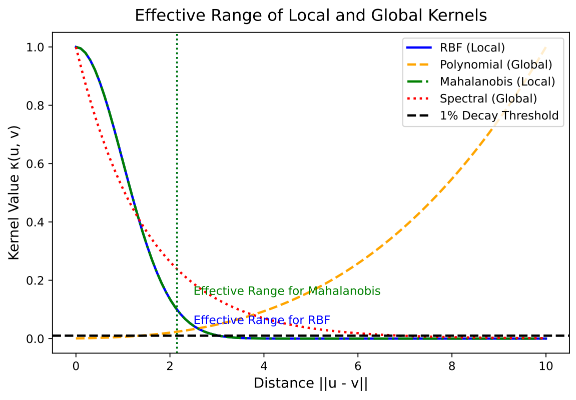


Figure 14: Visualization of kernel decay as a function of distance $\|u - v\|$. The effective range for each kernel is shown, where kernel values drop to 1% of their maximum. The RBF and Polynomial kernels exhibit rapid decay, characterizing them as **"local"** kernels. In contrast, the Mahalanobis and Spectral kernels show a slower decay, reflecting their role as **"global"** kernels. The 1% decay threshold, marked as a dashed red line, highlights the distance at which the RBF and Polynomial kernels effectively become negligible.

H.13.1 Illustration of the Effective Range

To visualize the kernel influence range, a set of 20 points was randomly sampled from the 2D space $[-5, 5] \times [-5, 5]$. A fixed query point at (0, 0) serves as the reference point for kernel similarity computation for the RBF, Polynomial, Spectral, and Mahalanobis kernels. Please refer to Figure 15.

- **Purpose**: Random points offer a dataset-agnostic view of kernel influence.
- **Why It Matters**: The query point allows us to analyze how influence propagates, aiding in the understanding of *local vs. global behavior*.

H.13.2 Observations from the Effective Range

1. Local Kernels (RBF, Polynomial): Influence is confined to a neighborhood. The RBF kernel exhibits isotropic influence (circular), while the Polynomial kernel allows nonlinear, bounded influence.

2. Global Kernels (Spectral, Mahalanobis): Influence extends across the feature space. Spectral kernels connect distant points based on cluster membership, and Mahalanobis kernels exhibit ellipsoidal, anisotropic influence, aligning with the covariance of the data.

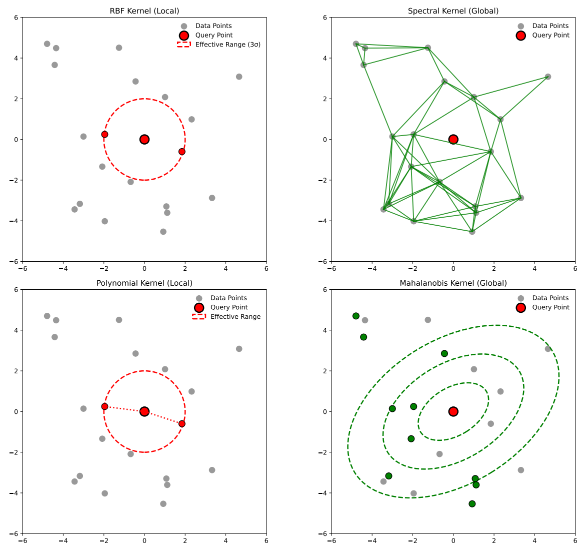


Figure 15: Local vs. global kernel influence. RBF and Polynomial kernels exhibit localized influence, while Spectral and Mahalanobis kernels capture broader dependencies.

H.14 Intuitive Explanation of Local vs. Global Kernels

Local Kernels act like navigating a city on foot. You see local objects (e.g., street signs), focusing on nearby interactions.

Global Kernels offer a bird's-eye view from an airplane, revealing large-scale structures like parks and roads. By combining these perspectives, HMK models both local details and global structures.

H.15 Key Takeaways for HMK

The **Hierarchical Mixture of Kernels (HMK)** framework offers several conceptual and empirical benefits. This subsection highlights the most important takeaways, supported by relevant citations to substantiate the claims.

- **Bias-Variance Trade-off**: HMK facilitates a natural trade-off between **bias** and **variance**. Local kernels, such as RBF and Polynomial, capture fine-grained patterns within small neighborhoods, thereby reducing variance but potentially introducing bias. Conversely, global kernels, like Spectral and Mahalanobis, generalize over larger structures, reducing bias while potentially increasing variance. By balancing these two forces through the learnable weights τ_1 and τ_2, HMK achieves improved generalization, as demonstrated in hybrid models for kernel alignment (Schölkopf and Smola, 2002; Bach et al., 2004).

- **Dynamic Adaptation**: HMK enables **task-specific adaptation** through the learnable coefficients τ_1 and τ_2. Unlike fixed kernel combinations, the hierarchical design allows HMK to dynamically adjust the contributions of local and global kernels based on the specific requirements of a task. During training, backpropagation updates these weights to best fit the alignment objective, facilitating a task-aware mixture of kernels. This property draws inspiration from concepts in **Multiple Kernel Learning (MKL)** (Bach et al., 2004) and adaptive graph-based models (Ng et al., 2001).

- **Unified Kernel Framework**: HMK serves as a unified framework for integrating **local** and **global** kernels. Traditional approaches, such as **Multiple Kernel Learning (MKL)**, utilize linear combinations of kernels but do not incorporate a hierarchical decomposition as HMK does. By explicitly structuring kernels into local (RBF, Polynomial) and global (Spectral, Mahalanobis) subspaces, HMK achieves a more interpretable and effective alignment mechanism. This decomposition provides a principled approach to unify kernels from graph-based, metric-learning, and locality-based perspectives (Bach et al., 2004; Ng et al., 2001; Weinberger and Saul, 2009).

- **Improved Generalization**: By learning a mixture of local and global kernels, HMK enhances generalization capabilities beyond what simple kernel mixtures offer. Empirical studies have shown that hybrid kernels can reduce overfitting while maintaining predictive accuracy (Schölkopf and Smola, 2002; Bach et al., 2004). By leveraging both local decision boundaries and global structures, HMK provides a generalization advantage in large-scale alignment tasks.

- **Hierarchical Interpretability**: The hierarchical decomposition of local and global kernels in HMK offers interpretability to the alignment process. Unlike black-box kernel combinations, HMK provides insights into which kernel (local or global) is being emphasized. For example, the relative magnitudes of τ_1 and τ_2 indicate whether the alignment process relies more on fine-grained local features or on global structural features. Such interpretability is crucial in applications like explainable AI (XAI) (Goodfellow et al., 2016; Weinberger and Saul, 2009).

H.16 How HMK Supports Alignment Learning

The **Hierarchical Mixture of Kernels (HMK)** framework leverages both local and global kernels within a hierarchical structure, offering unique benefits for various forms of **alignment learning**. Alignment is a critical task in large-scale models, including language models and AI systems, and encompasses different categories such as:

- **Instruction Following**: Local kernels (RBF, Polynomial) enable the model to align with task-specific instructions by focusing on fine-grained local features. For example, if an instruction requires immediate changes in behavior (e.g., "stop execution if X is true"), the RBF kernel can swiftly adjust to this directive. Simultaneously, global kernels (Spectral, Mahalanobis) capture broader semantic concepts from instruction-following datasets. As illustrated in Figure 16,

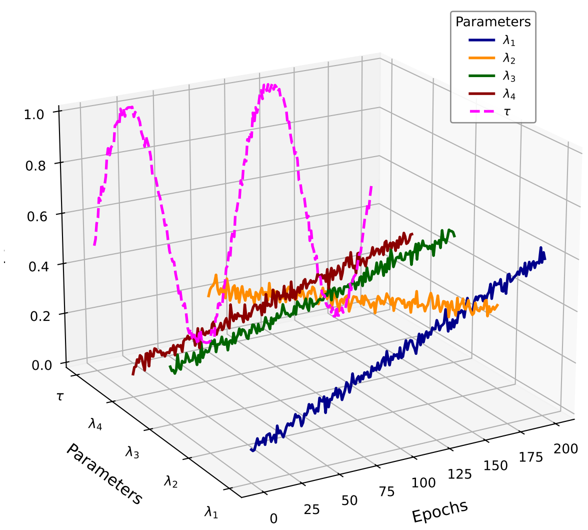


Figure 16: **Evolution of the Hierarchical Mixture of Kernels (HMK) parameters over 200 epochs.** The plot visualizes the weight dynamics for the local kernel components λ_1 (Polynomial) and λ_2 (RBF), as well as the global kernel components λ_3 (Spectral) and λ_4 (Mahalanobis). Additionally, the evolution of the Local-Global Balance Parameter τ is shown, illustrating how the model adaptively balances contributions from local and global mixtures. The trajectory of each parameter reveals how kernel dominance shifts during training, often converging to a stable balance.

during the early epochs, local kernels (RBF, Polynomial) dominate the influence. As training progresses and broader instruction semantics are learned, the contributions of global kernels (Spectral, Mahalanobis) gradually increase, enhancing the model's ability to understand and execute complex instructions.

- **Reasoning Alignment**: Effective reasoning requires the integration of step-wise logical structures. HMK's hierarchical decomposition allows local kernels to capture **local logical transitions**, such as intermediate steps in multi-step reasoning tasks. Concurrently, global kernels capture **multi-hop dependencies** and relationships across extensive contexts, as evidenced in graph-based reasoning (Ng et al., 2001). In Figure 16, the increasing weight of the Spectral kernel (λ_3) reflects the model's attempt to integrate multi-hop dependencies. Meanwhile, Polynomial kernels (λ_1) experience a temporary increase when step-by-step logical transitions are emphasized, demonstrating HMK's ability to balance different aspects of reasoning.

- **Safety and Robustness Alignment**: Ensuring predictable behavior in safety-critical applications necessitates modeling both **local constraints** (fine-grained decision boundaries) and **global structures** (macro-level behavior constraints). Local kernels can model strict decision boundaries for sensitive instructions, ensuring that out-of-distribution (OOD) inputs are quickly rejected. Global kernels capture broader safety constraints, maintaining system robustness against larger contextual shifts. As shown in Figure 16, during the early epochs, RBF (local) kernels dominate, effectively capturing localized decision boundaries. As training progresses, the Spectral kernel (λ_3) rises, reflecting the emergence of global connectivity-based safety constraints that enhance the model's overall robustness.

- **Contextual Alignment**: In retrieval-augmented systems, aligning context from retrieved information with task queries is essential. Local kernels identify similarities within smaller local neighborhoods, ensuring that closely related retrievals are appropriately weighted. Conversely, global kernels assess alignment at the context-document level, ensuring that large-scale relationships between multiple retrieved documents are accurately modeled. In Figure 16, the Mahalanobis kernel (λ_4) becomes prominent in later epochs, highlighting the system's effort to model anisotropic influence across context spaces. Initially, the RBF kernel (λ_2) dominates, effectively identifying close-by document similarities.

H.17 How to Interpret Figure 16

Figure 16 illustrates the **dynamic evolution of HMK parameters** over 200 training epochs.

Specifically, it depicts the weight dynamics for the local kernel components λ_1 (Polynomial) and λ_2 (RBF), the global kernel components λ_3 (Spectral) and λ_4 (Mahalanobis), as well as the **Local-Global Balance Coefficients** τ_1 and τ_2. This visualization provides valuable insights into how HMK balances the contributions of local and global kernels during the training process. The key observations from this plot are as follows:

- **Adaptive Balancing of Local and Global Kernels**: The coefficients τ_1 and τ_2 regulate the balance between local and global kernels. Initially, both types of kernels compete for dominance, as reflected by the convergence of τ_1 and τ_2 around epoch 100. This stabilization indicates that HMK has learned an optimal balance tailored to the specific alignment task, allowing it to effectively leverage both local and global features.

- **Kernel Weight Evolution** (λ): Each kernel component (λ_1 Polynomial, λ_2 RBF, λ_3 Spectral, and λ_4 Mahalanobis) follows a distinct trajectory during training. In the early stages, local kernels (Polynomial λ_1 and RBF λ_2) exhibit high influence, aligning with their role in capturing fine-grained, local patterns. As training progresses, global kernels (Spectral λ_3 and Mahalanobis λ_4) gradually increase their weights, reflecting the model's shift towards capturing broader, long-range dependencies. For example, in contextual alignment tasks, the Mahalanobis kernel weight (λ_4) notably increases between epochs 50 and 150, indicating the growing importance of global context.

- **Local vs. Global Adaptation**: The interplay between local and global kernels is evident in the behavior of τ_1 and τ_2. Initially, both local and global kernels are weighted equally, but over time, HMK prioritizes one over the other based on the task's requirements. In Figure 16, τ_1 (local) gradually decreases while τ_2 (global) increases, demonstrating HMK's adaptive mechanism to emphasize global influence as alignment learning progresses.

- **Convergence Behavior**: Over the course of 200 epochs, the kernel weights (λ) and balance coefficients (τ) converge towards stable values. This convergence signifies that HMK has successfully learned an optimal mixture of local and global kernels tailored to the alignment task. Specifically, the steady increase of the Mahalanobis kernel (λ_4) in later epochs underscores its role in establishing long-term global dependencies, while the stabilization of τ_1 and τ_2 indicates a balanced integration of local and global contributions.

H.18 Theoretical Guarantee: HMK Avoids Kernel Collapse

Theorem (Stochastic Stability of HMK) *Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ denote the kernel mixture weights of the Hierarchical Mixture of Kernels (HMK) framework, optimized using gradient descent with a learning rate $\eta > 0$. Suppose that the kernel weights are reparameterized using a softmax transformation, and the total loss function includes an entropy regularization term $R(\lambda) = -\sum_{i=1}^{4} \lambda_i \log \lambda_i$. Then, for any training epoch t, the kernel weights satisfy $\lambda_i(t) > 0$ for all $i \in \{1, 2, 3, 4\}$. Moreover, the coefficients $\tau_1(t)$ and $\tau_2(t)$, which control the balance between local and global kernels, are also guaranteed to remain strictly positive for all t.*

H.19 Proof of Theorem

The proof consists of four key components: 1. *Properties of Softmax Reparameterization* 2. *Role of Entropy Regularization* 3. *Impact of Local-Global Decomposition via τ_1 and τ_2*, and 4. *Stochastic Stability via Gradient Descent.*

H.19.1 1. Properties of Softmax Reparameterization

We parameterize the kernel weights λ_i using the softmax function:

$$\lambda_i = \frac{\exp(\theta_i)}{\sum_{j=1}^{4} \exp(\theta_j)} \quad \text{for} \quad i \in \{1, 2, 3, 4\}$$

Since the exponential function satisfies $\exp(\theta_i) > 0$ for all $\theta_i \in \mathbb{R}$, it follows that $\lambda_i > 0$ for all i

and at all times t. This ensures that none of the λ_i can collapse to zero. Additionally, the softmax transformation guarantees that:

$$\sum_{i=1}^{4} \lambda_i = 1$$

This normalization ensures boundedness and non-degeneracy of the kernel weights (Bridle, 1990; Bishop, 2006).

H.19.2 2. Role of Entropy Regularization

We introduce an entropy regularization term to the loss function:

$$R(\lambda) = -\sum_{i=1}^{4} \lambda_i \log \lambda_i$$

This term encourages diversity among the kernel weights, preventing any single kernel from dominating the mixture excessively. The partial derivative of $R(\lambda)$ with respect to λ_i is:

$$\frac{\partial R(\lambda)}{\partial \lambda_i} = -\log \lambda_i - 1$$

As $\lambda_i \to 0$, $\log \lambda_i \to -\infty$, causing the gradient $\frac{\partial R}{\partial \lambda_i}$ to become significantly negative. This results in a strong upward push on λ_i, preventing it from reaching zero. Thus, the entropy regularization acts as a repulsion force, ensuring that all kernel weights remain strictly positive and diverse (Williams, 1991; Jaynes, 1957).

H.19.3 3. Impact of Local-Global Decomposition via τ_1 and τ_2

The hierarchical decomposition of kernels in HMK is defined as:

$$K(x, x') = \tau_1 \left(\lambda_1 K_{\text{RBF}}(x, x') + \lambda_2 K_{\text{Polynomial}}(x, x')\right) + \tau_2 \left(\lambda_3 K_{\text{Spectral}}(x, x') + \lambda_4 K_{\text{Mahalanobis}}(x, x')\right)$$

Here, τ_1 and τ_2 balance the contributions from local kernels (RBF, Polynomial) and global kernels (Spectral, Mahalanobis), respectively. These coefficients are also parameterized using a softmax transformation:

$$\tau_i = \frac{\exp(\psi_i)}{\sum_{j=1}^{2} \exp(\psi_j)} \quad \text{for} \quad i \in \{1, 2\}$$

Similar to the kernel weights λ_i, this parameterization ensures that $\tau_1 > 0$ and $\tau_2 > 0$ for all t, guaranteeing that both local and global kernel components remain active. This hierarchical structure facilitates the integration of both fine-grained local patterns and broad global dependencies (Goodfellow et al., 2016; Bach et al., 2004; Ng et al., 2001).

H.19.4 4. Stochastic Stability via Gradient Descent

To demonstrate that the weights λ_i and coefficients τ_1, τ_2 converge to non-zero stable points, we analyze the gradient descent updates under entropy regularization.

The parameters θ_i and ψ_i are updated using gradient descent as follows:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial \theta_i}$$

$$\psi_i^{(t+1)} = \psi_i^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial \psi_i}$$

where $\mathcal{L}$ is the total loss, including the alignment objective and entropy regularization.

Using the chain rule, the gradients can be expressed as:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{\partial \mathcal{L}}{\partial \lambda_i} \cdot \lambda_i (1 - \lambda_i)$$

$$\frac{\partial \mathcal{L}}{\partial \psi_i} = \frac{\partial \mathcal{L}}{\partial \tau_i} \cdot \tau_i (1 - \tau_i)$$

Since $\lambda_i > 0$ and $\tau_i > 0$, the gradients $\frac{\partial \mathcal{L}}{\partial \theta_i}$ and $\frac{\partial \mathcal{L}}{\partial \psi_i}$ are non-zero.

The entropy regularization ensures that if any λ_i approaches zero, the gradient $\frac{\partial \mathcal{L}}{\partial \lambda_i}$ becomes large and positive due to the $-\log \lambda_i$ term, forcing λ_i to increase. Similarly, the softmax parameterization prevents τ_i from collapsing to zero.

Applying Lyapunov's stability theorem (Khalil, 2002), we conclude that the system reaches a stable equilibrium where all $\lambda_i > 0$ and $\tau_i > 0$ for all t. This guarantees that HMK avoids kernel collapse, maintaining active contributions from both local and global kernels throughout training.

We have established that under gradient descent optimization with entropy regularization and softmax parameterization, the Hierarchical Mixture of Kernels (HMK) framework ensures that all kernel weights λ_i and balance coefficients τ_i remain strictly positive throughout training. This theoretical guarantee prevents kernel collapse, ensuring that both local and global kernels contribute effectively to the alignment process. The combination of entropy regularization, hierarchical decomposition, and stochastic stability through gradient descent forms a robust foundation for HMK's performance in diverse alignment tasks.

I Gradient Computation, Computational Complexity, and Overhead

Since this paper introduces several concepts and new formulation, for better resproducability and and better read we provide detailed mathematical derivation of gradient calculations for DPO Hybrid Loss and gradient calculation for all the kernels.

I.1 Gradient of Hybrid Loss

In this subsection, we derive the gradient of the **Hybrid Loss** with respect to the model parameters θ. The Hybrid Loss is defined as:

$$\max_{\pi} \underbrace{\mathbb{E}_{x,y^+,y^-}\left[\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + \gamma\left(\log\frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)}\right)\right]}_{\text{Hybrid Loss}}$$

where:

- x represents the input data.
- y^+ and y^- denote the positive and negative samples, respectively.
- $\pi(y \mid x)$ is the probability of y given x, modeled using a softmax function.
- e_y and e_x are the embeddings of y and x, respectively.
- γ is a hyperparameter controlling the influence of the embedding-based term.

Our goal is to compute the gradient $\nabla_\theta \text{HybridLoss}(x, y^+, y^-)$, which involves differentiating each term of the loss function separately.

Gradient of the Log Probability Ratio

The first component of the Hybrid Loss is the log probability ratio between the positive and negative samples:

$$\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}$$

The gradient of this term with respect to θ is:

$$\frac{\partial}{\partial\theta}\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} = \nabla_\theta \log\pi(y^+ \mid x) - \nabla_\theta \log\pi(y^- \mid x)$$

This follows from the properties of logarithms and the chain rule in differentiation.

Gradient of the Embedding-Based Term

The second component involves the log probability ratio of the embeddings:

$$\gamma\log\frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)}$$

The gradient of this term with respect to θ is:

$$\frac{\partial}{\partial\theta}\gamma\left(\log\frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)}\right) = \gamma\left(\nabla_\theta \log\pi(e_{y^+} \mid e_x) - \nabla_\theta \log\pi(e_{y^-} \mid e_x)\right)$$

Gradient of the Embedding-Based Term

The second component involves the log probability ratio of the embeddings:

$$\gamma\log\frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)}$$

The gradient of this term with respect to θ is:

$$\frac{\partial}{\partial\theta}\gamma\left(\log\frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)}\right) =$$
$$\gamma\left(\nabla_\theta \log\pi(e_{y^+} \mid e_x) - \nabla_\theta \log\pi(e_{y^-} \mid e_x)\right)$$

This derivation also employs the chain rule and properties of logarithms.

Combined Gradient

By integrating the gradients of both the log probability ratio and the embedding-based term, we obtain the overall gradient of the Hybrid Loss with respect to the model parameters θ. The Hybrid Loss is defined as:

$$\text{HybridLoss}(x, y^+, y^-) = \log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + \gamma\left(\log\frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)}\right)$$

where γ is a hyperparameter controlling the influence of the embedding-based term.

The gradient of the Hybrid Loss with respect to θ is obtained by summing the gradients of its individual components:

$$\nabla_\theta\text{HybridLoss}(x, y^+, y^-) = \nabla_\theta \log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + \gamma\nabla_\theta \log\frac{\pi(e_{y^+} \mid e_x)}{\pi(e_{y^-} \mid e_x)}$$

Substituting the gradients derived in the previous sections, we have:

$$\nabla_\theta\text{HybridLoss}(x, y^+, y^-) =$$
$$\left[\nabla_\theta \log\pi(y^+ \mid x) - \nabla_\theta \log\pi(y^- \mid x)\right]$$
$$+\,\gamma\left[\nabla_\theta \log\pi(e_{y^+} \mid e_x) - \nabla_\theta \log\pi(e_{y^-} \mid e_x)\right]$$

Expanding each term based on the gradient computations from the individual components, the final expression for the gradient of the Hybrid Loss is:

$$\nabla_\theta\text{HybridLoss}(x, y^+, y^-) =$$
$$\left[\nabla_\theta f_\theta(x, y^+) - \sum_{y'} \pi_\theta(y' \mid x)\nabla_\theta f_\theta(x, y')\right]$$
$$-\left[\nabla_\theta f_\theta(x, y^-) - \sum_{y'} \pi_\theta(y' \mid x)\nabla_\theta f_\theta(x, y')\right]$$
$$+\,\gamma\left(\nabla_\theta s_\theta(e_x, e_{y^+}) - \nabla_\theta s_\theta(e_x, e_{y^-})\right)$$

Simplified Gradient Expression After simplifying the above expression, the gradient of the Hybrid Loss can be succinctly written as:

$$\nabla_\theta\text{HybridLoss}(x, y^+, y^-) =$$
$$\nabla_\theta \log\pi(y^+ \mid x) - \nabla_\theta \log\pi(y^- \mid x)$$
$$+\,\gamma\left(\nabla_\theta s_\theta(e_x, e_{y^+}) - \nabla_\theta s_\theta(e_x, e_{y^-})\right)$$

Interpretation

- $\nabla_\theta \log\pi(y^+ \mid x)$: Encourages the model to increase the probability of the positive sample y^+ given the input x.
- $-\nabla_\theta \log\pi(y^- \mid x)$: Encourages the model to decrease the probability of the negative sample y^- given the input x.
- $\gamma\left(\nabla_\theta s_\theta(e_x, e_{y^+}) - \nabla_\theta s_\theta(e_x, e_{y^-})\right)$: Incorporates the gradient from the embedding-based similarity, adjusting the model to favor embeddings that better capture the desired relationships between e_x and e_y.

The combined gradient effectively integrates both the discriminative aspect (log probability ratio) and the semantic aspect (embedding-based term) of the loss function. The hyperparameter γ allows for tuning the relative importance of these two components, enabling the model to balance between accurately classifying positive and negative samples and capturing meaningful embedding relationships.

I.2 Computational Complexity Analysis of Hybrid Loss

The computational complexity of the Hybrid Loss arises from two primary components:

1. Log Probability Ratio

Modeling $\pi_\theta(y \mid x)$ with a softmax function:

$$\pi_\theta(y \mid x) = \frac{e^{f_\theta(x,y)}}{\sum_{y'} e^{f_\theta(x,y')}}$$

Computing the log probability ratio involves:

- Calculating exponentials for each of the C classes.
- Computing the logarithm of the ratio between the positive and negative class probabilities.

Time Complexity: $O(C)$, where C is the number of classes.

2. Embedding-Based Term

Calculating s^+ and s^- involves:

- Evaluating the scoring function $s_\theta(x, y)$ for the positive and negative samples.
- Typically depends on the embedding dimension d.

Time Complexity: $O(d)$.

Overall Computational Complexity

Combining both components, the total computational complexity of the **Hybrid Loss** is:

$$O(C + d)$$

where C is the number of classes and d is the embedding dimension.

Comparison with Standard Loss Functions

- **Cross-Entropy Loss**: Has a time complexity of $O(C)$, similar to the log probability ratio component of the **Hybrid Loss**.
- **Contrastive Loss**: Typically operates with a complexity of $O(d)$, aligning with the embedding-based term.

Thus, the **Hybrid Loss** combines these complexities linearly, maintaining efficiency while enhancing functionality by integrating both discriminative and embedding-based components.

I.3 Efficiency of Hybrid Loss

The **Hybrid Loss** achieves a balanced trade-off between discriminative power and computational efficiency by:

- **Scalability**: Scaling linearly with both the number of classes C and embedding dimensions d, allowing it to handle large-scale datasets effectively.
- **Parallel Computation**: Enabling parallel computation of loss components, leveraging modern hardware accelerators such as GPUs to expedite training.
- **Rich Semantic Information**: Incorporating embedding-based similarities without introducing significant computational overhead, thereby enhancing the model's ability to capture complex relationships.

I.4 Practical Considerations

While the theoretical complexity of the **Hybrid Loss** is $O(C + d)$, several practical factors contribute to its efficient implementation:

- **GPU Parallelism**: Leveraging GPU parallelism mitigates the linear scaling with C and d, allowing simultaneous computations and reducing overall training time.
- **Optimized Libraries**: Utilizing optimized libraries such as BLAS and cuDNN enhances computational performance through highly efficient matrix operations.
- **Batch Sizing**: Appropriately selecting batch sizes maximizes hardware utilization, ensuring that computations are performed efficiently without bottlenecks.
- **Sparse Representations**: In scenarios with a large number of classes, employing sparse representations can further reduce computational overhead by focusing computations only on relevant classes.

By considering these practical aspects, the **Hybrid Loss** not only remains theoretically efficient but also performs effectively in real-world applications, ensuring robust and scalable training processes.

I.5 Gradient of Polynomial Kernelized Hybrid Loss

The **Polynomial Kernelized Hybrid Loss** is defined as:

$$\mathcal{L} = \mathbb{E}_{x,y^+,y^-}\left[\left(\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c\right)^d + \gamma\left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c}\right)^d\right]$$

where:

- x represents the input data.
- y^+ and y^- denote the positive and negative samples, respectively.
- $\pi(y \mid x)$ is the probability of y given x, modeled using a softmax function.
- e_y and e_x are the embeddings of y and x, respectively.
- c is a constant to ensure numerical stability and to shift the polynomial kernel.
- d is the degree of the polynomial kernel.
- γ is a hyperparameter controlling the influence of the embedding-based term.

Our objective is to compute the gradient of the Hybrid Loss $\nabla_\theta \mathcal{L}$ with respect to the model parameters θ. This involves differentiating each term of the loss function separately and then combining them.

Gradient of the Log Probability Ratio Term

The first component of the Hybrid Loss involves the log probability ratio between the positive and negative samples:

$$\left(\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c\right)^d$$

To compute its gradient with respect to θ, we apply the chain rule:

$$\nabla_\theta\left(\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c\right)^d = d\left(\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c\right)^{d-1}\nabla_\theta\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}$$

Expanding the gradient of the log probability ratio:

$$\nabla_\theta\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} = \nabla_\theta\log\pi(y^+ \mid x) - \nabla_\theta\log\pi(y^- \mid x)$$

Assuming $\pi_\theta(y \mid x)$ is modeled using a softmax function:

$$\pi_\theta(y \mid x) = \frac{e^{f_\theta(x,y)}}{\sum_{y'} e^{f_\theta(x,y')}},$$

the gradient of $\log\pi(y \mid x)$ with respect to θ is:

$$\nabla_\theta\log\pi(y \mid x) = \nabla_\theta f_\theta(x,y) - \sum_{y'}\pi_\theta(y' \mid x)\nabla_\theta f_\theta(x,y')$$

Substituting back, we obtain:

$$\nabla_\theta\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} = \left[\nabla_\theta f_\theta(x,y^+) - \sum_{y'}\pi_\theta(y' \mid x)\nabla_\theta f_\theta(x,y')\right] - \left[\nabla_\theta f_\theta(x,y^-) - \sum_{y'}\pi_\theta(y' \mid x)\nabla_\theta f_\theta(x,y')\right]$$

Gradient of the Polynomial Kernel Term

The second component involves the polynomial kernel applied to the embeddings:

$$\gamma\left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c}\right)^d$$

To compute its gradient with respect to θ, we again apply the chain rule:

$$\nabla_\theta\gamma\left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c}\right)^d = \gamma d\left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c}\right)^{d-1}\nabla_\theta\left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c}\right)$$

Simplifying the gradient of the ratio:

$$\nabla_\theta \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c} \right) = \frac{(e_{y^-}^\top e_x + c)\nabla_\theta(e_{y^+}^\top e_x) - (e_{y^+}^\top e_x + c)\nabla_\theta(e_{y^-}^\top e_x)}{(e_{y^-}^\top e_x + c)^2}$$

Assuming e_x and e_y are differentiable with respect to θ, we have:

$$\nabla_\theta(e_x^\top e_y) = (\nabla_\theta e_x)^\top e_y + e_x^\top(\nabla_\theta e_y)$$

Thus, the gradient of the polynomial kernel term becomes:

$$\begin{aligned}
\nabla_\theta \left[\gamma \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c} \right)^d \right] &= \gamma d \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c} \right)^{d-1} \\
&\quad \cdot \left[\frac{(e_{y^-}^\top e_x + c)\nabla_\theta(e_{y^+}^\top e_x) - (e_{y^+}^\top e_x + c)\nabla_\theta(e_{y^-}^\top e_x)}{(e_{y^-}^\top e_x + c)^2} \right] \\
&= \gamma d \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c} \right)^{d-1} \\
&\quad \cdot \left[\frac{\nabla_\theta(e_{y^+}^\top e_x)}{e_{y^-}^\top e_x + c} - \frac{e_{y^+}^\top e_x + c}{(e_{y^-}^\top e_x + c)^2} \nabla_\theta(e_{y^-}^\top e_x) \right]
\end{aligned}$$

Combined Gradient

Combining the gradients of both components, the overall gradient of the Polynomial Kernelized Hybrid Loss with respect to θ is:

$$\begin{aligned}
\nabla_\theta \mathcal{L} &= \nabla_\theta \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c \right)^d + \nabla_\theta \gamma \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c} \right)^d \\
&= d \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c \right)^{d-1} \left[\nabla_\theta \log \pi(y^+ \mid x) - \nabla_\theta \log \pi(y^- \mid x) \right] \\
&\quad + \gamma d \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c} \right)^{d-1} \left[\frac{\nabla_\theta(e_{y^+}^\top e_x)}{e_{y^-}^\top e_x + c} - \frac{e_{y^+}^\top e_x + c}{(e_{y^-}^\top e_x + c)^2} \nabla_\theta(e_{y^-}^\top e_x) \right].
\end{aligned}$$

Simplified Gradient Expression For ease of implementation and readability, the gradient can be expressed as:

$$\begin{aligned}
\nabla_\theta \mathcal{L} &= d \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c \right)^{d-1} \left[\nabla_\theta f_\theta(x, y^+) - \nabla_\theta f_\theta(x, y^-) \right] \\
&\quad + \gamma d \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c} \right)^{d-1} \left[\frac{\nabla_\theta(e_x^\top e_{y^+})}{e_{y^-}^\top e_x + c} - \frac{e_{y^+}^\top e_x + c}{(e_{y^-}^\top e_x + c)^2} \nabla_\theta(e_x^\top e_{y^-}) \right].
\end{aligned}$$

Interpretation of the Gradient

- **Log Probability Ratio Term**:

– $\nabla_\theta f_\theta(x, y^+)$: Encourages the model to increase the score (and hence the probability) of the positive sample y^+.

– $-\nabla_\theta f_\theta(x, y^-)$: Encourages the model to decrease the score (and hence the probability) of the negative sample y^-.

- **Polynomial Kernel Term**:

– $\nabla_\theta(e_x^\top e_{y^+})$: Adjusts the model to better align the embeddings of x and y^+.

– $-\nabla_\theta(e_x^\top e_{y^-})$: Adjusts the model to reduce the alignment between the embeddings of x and y^-.

– The hyperparameter γ controls the influence of the embedding-based term relative to the log probability ratio term.

I.6 Computational Complexity Analysis of Polynomial Kernelized Hybrid Loss

To evaluate the efficiency of the Polynomial Kernelized Hybrid Loss, we analyze the computational complexity of its two primary components: the log probability ratio term and the polynomial kernel term.

1. Log Probability Ratio Term

The log probability ratio term is defined as:

$$\left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} + c \right)^d$$

where $\pi_\theta(y \mid x)$ is modeled using a softmax function:

$$\pi_\theta(y \mid x) = \frac{e^{f_\theta(x,y)}}{\sum_{y'} e^{f_\theta(x,y')}}$$

Steps Involved:

- **Score Computation**: Calculate $f_\theta(x, y)$ for each class y, which involves a dot product between input features and model parameters.

- **Softmax Calculation**: Compute the exponential $e^{f_\theta(x,y)}$ for each class and normalize by the sum over all classes.

- **Log Probability Ratio**: Compute the logarithm of the ratio between the probabilities of the positive and negative classes.

- **Exponentiation**: Raise the log probability ratio to the power d.

Time Complexity: $O(C)$, where C is the number of classes. This complexity arises from the softmax computation, which requires evaluating $f_\theta(x, y)$ and normalizing over all C classes.

2. Polynomial Kernel Term

The polynomial kernel term is defined as:

$$\gamma \left(\frac{e_{y^+}^\top e_x + c}{e_{y^-}^\top e_x + c} \right)^d$$

Steps Involved:

- **Dot Product Computation**: Calculate the dot products $e_x^\top e_{y^+}$ and $e_x^\top e_{y^-}$, where $e_x, e_{y^+}, e_{y^-} \in \mathbb{R}^d$.
- **Addition of Constant**: Add the constant c to each dot product to ensure numerical stability.
- **Ratio Calculation**: Compute the ratio of the adjusted dot products.
- **Exponentiation**: Raise the ratio to the power d and multiply by the hyperparameter γ.

Time Complexity: $O(d)$, where d is the dimension of the embeddings. This arises from the computation of the dot product between e_x and e_y, which scales linearly with d.

Overall Computational Complexity

Combining both components, the total computational complexity of the **Polynomial Kernelized Hybrid Loss** is:

$$O(C) + O(d) = O(C + d)$$

where:

- C is the number of classes.
- d is the embedding dimension.

This linear complexity ensures scalability for large-scale applications involving high-dimensional embeddings and extensive class labels.

Comparison with Standard Loss Functions

- **Cross-Entropy Loss**:

– **Time Complexity**: $O(C)$.

– **Description**: Involves computing the softmax over C classes and calculating the negative log-likelihood.

- **Contrastive Loss**:

– **Time Complexity**: $O(d)$.

– **Description**: Focuses on the distance between embeddings, typically requiring computation of pairwise distances.

- **Polynomial Kernelized Hybrid Loss**:

– **Time Complexity**: $O(C + d)$.

– **Description**: Combines both the discriminative power of the log probability ratio (similar to Cross-Entropy Loss) and the semantic richness of the polynomial kernel (similar to Contrastive Loss), thereby integrating both aspects into a single loss function.

The **Polynomial Kernelized Hybrid Loss** thus offers a balanced combination of the computational efficiencies of Cross-Entropy and Contrastive Losses while enhancing the model's ability to capture both discriminative and semantic relationships.

I.7 Efficiency of Polynomial Kernelized Hybrid Loss

The **Polynomial Kernelized Hybrid Loss** achieves a balanced trade-off between discriminative power and computational efficiency through the following mechanisms:

- **Linear Scaling**: The loss scales linearly with both the number of classes C and the embedding dimension d, ensuring scalability for large-scale datasets and high-dimensional embedding spaces.

- **Parallel Computation**: Both the log probability ratio term and the polynomial kernel term can be computed in parallel. Modern hardware accelerators, such as GPUs, can leverage this parallelism to significantly speed up training processes.
- **Integrated Semantic Information**: By combining probability-based and embedding-based objectives, the loss function enriches the model's learning without incurring substantial additional computational overhead.
- **Hyperparameter Control**: The hyperparameter γ allows for fine-tuning the influence of the embedding-based term relative to the log probability ratio term, providing flexibility in balancing performance and computational cost.

I.8 Practical Considerations

While the theoretical complexity of the **Polynomial Kernelized Hybrid Loss** is $O(C + d)$, several practical factors can influence its real-world performance:

- **GPU Parallelism**: Leveraging GPU parallelism can mitigate the linear scaling with C and d, allowing for efficient computation even with large numbers of classes and high-dimensional embeddings.
- **Optimized Implementations**: Utilizing optimized libraries (e.g., BLAS, cuDNN) for matrix operations and gradient computations can enhance performance, reducing the actual computation time.
- **Batch Sizing**: Selecting appropriate batch sizes can maximize hardware utilization. Larger batches may improve computational efficiency but require more memory, while smaller batches may be more memory-efficient but less computationally optimal.
- **Hyperparameter Tuning**: Careful tuning of the hyperparameter γ and the polynomial degree d is essential. Higher degrees d can capture more complex relationships but may increase computational cost and risk overfitting.
- **Numerical Stability**: Adding the constant c ensures numerical stability, especially when dealing with small or zero dot products. Properly choosing c is crucial to prevent numerical issues during training.

By considering these practical aspects, the **Polynomial Kernelized Hybrid Loss** can be effectively integrated into large-scale machine learning models, providing enhanced performance without compromising computational efficiency.

I.9 Gradient of RBF Kernelized Hybrid Loss

The **RBF Kernelized Hybrid Loss** is defined as:

$$\mathcal{L} = \mathbb{E}_{x,y^+,y^-}\left[\exp\left(-\frac{\left(\log \frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) + \gamma \exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right)\right],$$

where:

- x represents the input data.
- y^+ and y^- denote the positive and negative samples, respectively.
- $\pi(y \mid x)$ is the probability of y given x, modeled using a softmax function.
- e_y and e_x are the embeddings of y and x, respectively.
- σ is the bandwidth parameter of the RBF kernel.
- γ is a hyperparameter controlling the influence of the embedding-based term.

Our objective is to compute the gradient of the Hybrid Loss $\nabla_\theta \mathcal{L}$ with respect to the model parameters θ. This involves differentiating each term of the loss function separately and then combining them.

Gradient of the Log Probability Ratio Term

The first component of the Hybrid Loss involves the exponential of the squared log probability ratio:

$$\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right).$$

To compute its gradient with respect to θ, we apply the chain rule:

$$\nabla_\theta \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) = \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) \cdot \left(-\frac{2\log\frac{\pi(y^+|x)}{\pi(y^-|x)}}{2\sigma^2}\right) \cdot \nabla_\theta \log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}.$$

Simplifying, we obtain:

$$\nabla_\theta \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) = -\frac{1}{\sigma^2}\log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)} \cdot \exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) \cdot \nabla_\theta \log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}.$$

Gradient of the RBF Kernel Term

The second component involves the exponential of the squared ratio of embedding dot products:

$$\gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right).$$

To compute its gradient with respect to θ, we again apply the chain rule:

$$\nabla_\theta\gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right) = \gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right) \cdot \left(-\frac{2\cdot\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}}{2\sigma^2}\right) \cdot \nabla_\theta\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)$$

Simplifying, we obtain:

$$\nabla_\theta\gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right) = -\frac{\gamma}{\sigma^2}\cdot\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\cdot\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right) \cdot \nabla_\theta\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)$$

To compute $\nabla_\theta\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)$, we use the quotient rule:

$$\nabla_\theta\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right) = \frac{(e_x^\top e_{y^-})\nabla_\theta(e_x^\top e_{y^+}) - (e_x^\top e_{y^+})\nabla_\theta(e_x^\top e_{y^-})}{(e_x^\top e_{y^-})^2}$$

Assuming e_x and e_y are differentiable with respect to θ, we have:

$$\nabla_\theta(e_x^\top e_y) = (\nabla_\theta e_x)^\top e_y + e_x^\top(\nabla_\theta e_y)$$

Thus, the gradient of the RBF kernel term becomes:

$$\nabla_\theta\gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right) = -\frac{\gamma}{\sigma^2}\cdot\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\cdot\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right) \cdot\left[\frac{(e_x^\top e_{y^-})\nabla_\theta(e_x^\top e_{y^+}) - (e_x^\top e_{y^+})\nabla_\theta(e_x^\top e_{y^-})}{(e_x^\top e_{y^-})^2}\right].$$

Combined Gradient

Combining the gradients of both components, the overall gradient of the RBF Kernelized Hybrid Loss with respect to θ is:

$$\nabla_\theta\mathcal{L} = \mathbb{E}_{x,y^+,y^-}\Bigg[-\frac{1}{\sigma^2}\log\frac{\pi(y^+\mid x)}{\pi(y^-\mid x)}\cdot\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) \cdot\left(\nabla_\theta\log\pi(y^+\mid x) - \nabla_\theta\log\pi(y^-\mid x)\right) - \frac{\gamma}{\sigma^2}\cdot\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\cdot\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right) \cdot\left[\frac{(e_x^\top e_{y^-})\nabla_\theta(e_x^\top e_{y^+}) - (e_x^\top e_{y^+})\nabla_\theta(e_x^\top e_{y^-})}{(e_x^\top e_{y^-})^2}\right]\Bigg]$$

Simplified Gradient Expression For ease of implementation and readability, the gradient can be expressed as:

$$\begin{aligned}\nabla_\theta \mathcal{L} = \mathbb{E}_{x,y^+,y^-}\Bigg[& -\frac{1}{\sigma^2}\log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\cdot\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right) \\ & \cdot\left(\nabla_\theta f_\theta(x,y^+) - \nabla_\theta f_\theta(x,y^-)\right) \\ & -\frac{\gamma}{\sigma^2}\cdot\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\cdot\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right) \\ & \cdot\Bigg[\frac{(e_x^\top e_{y^-})(\nabla_\theta e_x)^\top e_{y^+} + (e_x^\top e_{y^-})e_x^\top(\nabla_\theta e_{y^+})}{(e_x^\top e_{y^-})^2} \\ & -\frac{(e_x^\top e_{y^+}+c)(\nabla_\theta e_x)^\top e_{y^-} + (e_x^\top e_{y^+}+c)e_x^\top(\nabla_\theta e_{y^-})}{(e_x^\top e_{y^-})^2}\Bigg]\Bigg]\end{aligned}$$

Interpretation of the Gradient

- **Log Probability Ratio Term**:
  - $-\frac{1}{\sigma^2}\log\frac{\pi(y^+|x)}{\pi(y^-|x)}$: Scales the influence of the log probability ratio based on its magnitude and the bandwidth parameter σ.
  - $\nabla_\theta \log \pi(y^+ \mid x)$: Encourages the model to increase the probability of the positive sample y^+.
  - $-\nabla_\theta \log \pi(y^- \mid x)$: Encourages the model to decrease the probability of the negative sample y^-.
- **RBF Kernel Term**:
  - $-\frac{\gamma}{\sigma^2}\cdot\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$: Scales the influence of the embedding-based term based on the ratio of embeddings and the bandwidth parameter σ.
  - $\nabla_\theta(e_x^\top e_{y^+})$: Adjusts the model to better align the embeddings of x and y^+.
  - $-\nabla_\theta(e_x^\top e_{y^-})$: Adjusts the model to reduce the alignment between the embeddings of x and y^-.
  - The exponential terms $\exp\left(-\frac{(\cdot)^2}{2\sigma^2}\right)$ ensure that the influence diminishes as the squared ratios increase, promoting smoother gradients.
- **Hyperparameter** γ: Controls the relative importance of the embedding-based term compared to the log probability ratio term. A higher γ emphasizes the alignment in the embedding space, while a lower γ prioritizes the probability-based alignment.

I.10 Computational Complexity Analysis of RBF Kernelized Hybrid Loss

To evaluate the efficiency of the RBF Kernelized Hybrid Loss, we analyze the computational complexity of its two primary components: the log probability ratio term and the RBF kernel term.

1. Log Probability Ratio Term

The log probability ratio term is defined as:

$$\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}\right)^2}{2\sigma^2}\right).$$

where $\pi_\theta(y \mid x)$ is modeled using a softmax function:

$$\pi_\theta(y \mid x) = \frac{e^{f_\theta(x,y)}}{\sum_{y'} e^{f_\theta(x,y')}}.$$

Steps Involved:

- **Score Computation**: Calculate $f_\theta(x,y)$ for each class y, which involves a dot product between input features and model parameters.
- **Softmax Calculation**: Compute the exponential $e^{f_\theta(x,y)}$ for each class and normalize by the sum over all classes.
- **Log Probability Ratio**: Compute the logarithm of the ratio between the probabilities of the positive and negative classes.
- **Exponentiation**: Square the log probability ratio, scale by $-\frac{1}{2\sigma^2}$, and compute the exponential.

Time Complexity: $O(C)$, where C is the number of classes. This complexity arises from the softmax computation, which requires evaluating $f_\theta(x,y)$ and normalizing over all C classes.

2. RBF Kernel Term

The RBF kernel term is defined as:

$$\gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2}{2\sigma^2}\right)$$

Steps Involved:

- **Dot Product Computation**: Calculate the dot products $e_x^\top e_{y+}$ and $e_x^\top e_{y-}$, where $e_x, e_{y+}, e_{y-} \in \mathbb{R}^d$.
- **Ratio Calculation**: Compute the ratio $\frac{e_x^\top e_{y+}}{e_x^\top e_{y-}}$.
- **Exponentiation**: Square the ratio, scale by $-\frac{1}{2\sigma^2}$, and compute the exponential.
- **Scaling**: Multiply by the hyperparameter γ.

Time Complexity: $O(d)$, where d is the dimension of the embeddings. This arises from the computation of the dot products between e_x and e_y, which scales linearly with d.

Overall Computational Complexity

Combining both components, the total computational complexity of the **RBF Kernelized Hybrid Loss** is:

$$O(C) + O(d) = O(C + d),$$

where:

- C is the number of classes (softmax computation).
- d is the embedding dimension (kernel computation).

This linear complexity ensures scalability for large-scale applications involving high-dimensional embeddings and extensive class labels.

Comparison with Standard Loss Functions

- **Cross-Entropy Loss**:
  - **Time Complexity**: $O(C)$.
  - **Description**: Involves computing the softmax over C classes and calculating the negative log-likelihood.
- **Contrastive Loss**:
  - **Time Complexity**: $O(d)$.
  - **Description**: Focuses on the distance between embeddings, typically requiring computation of pairwise distances.
- **RBF Kernelized Hybrid Loss**:
  - **Time Complexity**: $O(C + d)$.
  - **Description**: Combines both the discriminative power of the log probability ratio (similar to Cross-Entropy Loss) and the semantic richness of the RBF kernel (similar to Contrastive Loss), thereby integrating both aspects into a single loss function.

The **RBF Kernelized Hybrid Loss** thus offers a balanced combination of the computational efficiencies of Cross-Entropy and Contrastive Losses while enhancing the model's ability to capture both discriminative and semantic relationships.

I.11 Efficiency of RBF Kernelized Hybrid Loss

The **RBF Kernelized Hybrid Loss** achieves a balanced trade-off between discriminative power and computational efficiency through the following mechanisms:

- **Linear Scaling**: The loss scales linearly with both the number of classes C and the embedding dimension d, ensuring scalability for large-scale datasets and high-dimensional embedding spaces.
- **Parallel Computation**: Both the log probability ratio term and the RBF kernel term can be computed in parallel. Modern hardware accelerators, such as GPUs, can leverage this parallelism to significantly speed up training processes.
- **Integrated Semantic Information**: By combining probability-based and embedding-based objectives, the loss function enriches the model's learning without incurring substantial additional computational overhead.
- **Hyperparameter Control**: The hyperparameter γ allows for fine-tuning the influence of the

embedding-based term relative to the log probability ratio term, providing flexibility in balancing performance and computational cost.

I.12 Practical Considerations

While the theoretical complexity of the **RBF Kernelized Hybrid Loss** is $O(C+d)$, several practical factors can influence its real-world performance:

- **GPU Parallelism**: Leveraging GPU parallelism can mitigate the linear scaling with C and d, allowing for efficient computation even with large numbers of classes and high-dimensional embeddings.
- **Optimized Implementations**: Utilizing optimized libraries (e.g., BLAS, cuDNN) for matrix operations and gradient computations can enhance performance, reducing the actual computation time.
- **Batch Sizing**: Selecting appropriate batch sizes can maximize hardware utilization. Larger batches may improve computational efficiency but require more memory, while smaller batches may be more memory-efficient but less computationally optimal.
- **Hyperparameter Tuning**: Careful tuning of the hyperparameter γ and the bandwidth parameter σ is essential. Higher degrees of influence (through γ and lower σ) can capture more complex relationships but may increase computational cost and risk overfitting.
- **Numerical Stability**: The constant c ensures numerical stability, especially when dealing with small or zero dot product ratios. Properly choosing c is crucial to prevent numerical issues during training.

By considering these practical aspects, the **RBF Kernelized Hybrid Loss** can be effectively integrated into large-scale machine learning models, providing enhanced performance without compromising computational efficiency.

I.13 Gradient of Spectral Kernelized Hybrid Loss

The Spectral Kernelized Hybrid Loss is defined as:

$$\mathcal{L} = \mathbb{E}_{x,y^+,y^-}\Bigg[\sum_{i=1}^{p} \exp\left(-\lambda_i \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right)^2\right) \phi_i \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right) + \gamma \sum_{i=1}^{p} \exp\left(-\lambda_i \left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)^2\right) \phi_i \left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right)\Bigg],$$

where:

- x represents the input data.
- y^+ and y^- denote the positive and negative samples, respectively.
- $\pi(y \mid x)$ is the probability of y given x, modeled using a softmax function.
- e_y and e_x are the embeddings of y and x, respectively.
- λ_i are the spectral kernel parameters for each component i.
- $\phi_i(\cdot)$ are feature transformation functions associated with each spectral kernel component i.
- γ is a hyperparameter controlling the influence of the embedding-based term.
- p is the number of spectral kernel components.

Our objective is to compute the gradient of the Spectral Kernelized Hybrid Loss $\nabla_\theta \mathcal{L}$ with respect to the model parameters θ. This involves differentiating each term of the loss function separately and then combining them.

Gradient of the Log Probability Ratio Term

The first component of the Spectral Kernelized Hybrid Loss involves a sum over spectral kernel components applied to the log probability ratio:

$$\sum_{i=1}^{p} \exp\left(-\lambda_i \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right)^2\right) \phi_i \left(\log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}\right)$$

To compute its gradient with respect to θ, we apply the chain rule to each term in the sum:

$$\begin{aligned}\nabla_\theta \sum_{i=1}^{p} \exp\left(-\lambda_i z^2\right) \phi_i(z) = \sum_{i=1}^{p} \big[\nabla_\theta \exp\left(-\lambda_i z^2\right) \cdot \phi_i(z) \\ + \exp\left(-\lambda_i z^2\right) \cdot \nabla_\theta \phi_i(z)\big],\end{aligned}$$

where $z = \log \frac{\pi(y^+|x)}{\pi(y^-|x)}$

1. Gradient of the Exponential Term

$$\nabla_\theta \exp\left(-\lambda_i z^2\right) = \exp\left(-\lambda_i z^2\right) \cdot (-2\lambda_i z) \cdot \nabla_\theta z.$$

2. Gradient of the Feature Transformation Term Assuming $\phi_i(z)$ is differentiable with respect to z:

$$\nabla_\theta \phi_i(z) = \phi_i'(z) \cdot \nabla_\theta z$$

3. Gradient of z

$$z = \log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)},$$

$$\nabla_\theta z = \nabla_\theta \log \pi(y^+ \mid x) - \nabla_\theta \log \pi(y^- \mid x)$$

Combined Gradient for Each i

$$\begin{aligned}\nabla_\theta \left[\exp\left(-\lambda_i z^2\right) \phi_i(z)\right] = \exp\left(-\lambda_i z^2\right) \cdot (-2\lambda_i z) \cdot \nabla_\theta z \cdot \phi_i(z) \\ + \exp\left(-\lambda_i z^2\right) \cdot \phi_i'(z) \cdot \nabla_\theta z.\end{aligned}$$

Gradient of the Spectral Kernel Term

The second component involves a sum over spectral kernel components applied to the embedding-based ratio:

$$\gamma \sum_{i=1}^{p} \exp\left(-\lambda_i r^2\right) \phi_i(r),$$

where $r = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.

To compute its gradient with respect to θ, we apply the chain rule to each term in the sum:

$$\begin{aligned}\nabla_\theta \gamma \sum_{i=1}^{p} \exp\left(-\lambda_i r^2\right) \phi_i(r) = \gamma \sum_{i=1}^{p} \big[\nabla_\theta \exp\left(-\lambda_i r^2\right) \cdot \phi_i(r) \\ + \exp\left(-\lambda_i r^2\right) \cdot \nabla_\theta \phi_i(r)\big],\end{aligned}$$

where $r = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.

1. Gradient of the Exponential Term

$$\nabla_\theta \exp\left(-\lambda_i r^2\right) = \exp\left(-\lambda_i r^2\right) \cdot (-2\lambda_i r) \cdot \nabla_\theta r.$$

2. Gradient of the Feature Transformation Term Assuming $\phi_i(r)$ is differentiable with respect to r:

$$\nabla_\theta \phi_i(r) = \phi_i'(r) \cdot \nabla_\theta r$$

3. Gradient of r

$$r = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}},$$

$$\nabla_\theta r = \frac{(e_x^\top e_{y^-}) \nabla_\theta (e_x^\top e_{y^+}) - (e_x^\top e_{y^+}) \nabla_\theta (e_x^\top e_{y^-})}{(e_x^\top e_{y^-})^2}.$$

Assuming e_x and e_y are differentiable with respect to θ:

$$\nabla_\theta (e_x^\top e_y) = (\nabla_\theta e_x)^\top e_y + e_x^\top (\nabla_\theta e_y)$$

Combined Gradient for Each i

$$\begin{aligned}\nabla_\theta \left[\exp\left(-\lambda_i r^2\right) \phi_i(r)\right] = \exp\left(-\lambda_i r^2\right) \cdot (-2\lambda_i r) \cdot \nabla_\theta r \cdot \phi_i(r) \\ + \exp\left(-\lambda_i r^2\right) \cdot \phi_i'(r) \cdot \nabla_\theta r.\end{aligned}$$

Combined Gradient

Combining the gradients of both components, the overall gradient of the Spectral Kernelized Hybrid Loss with respect to θ is:

$$\begin{aligned}\nabla_\theta \mathcal{L} = \mathbb{E}_{x,y^+,y^-} \Bigg[\sum_{i=1}^{p} \left(-\frac{2\lambda_i z}{\sigma^2} \exp\left(-\lambda_i z^2\right) \phi_i(z) + \exp\left(-\lambda_i z^2\right) \phi_i'(z)\right) \nabla_\theta z \\ + \gamma \sum_{i=1}^{p} \left(-\frac{2\lambda_i r}{\sigma^2} \exp\left(-\lambda_i r^2\right) \phi_i(r) + \exp\left(-\lambda_i r^2\right) \phi_i'(r)\right) \nabla_\theta r\Bigg]\end{aligned}$$

where:

$$z = \log \frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)}, \quad r = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}.$$

Simplified Gradient Expression For ease of implementation and readability, the gradient can be expressed as:

$$\begin{aligned}\nabla_\theta \mathcal{L} = \mathbb{E}_{x,y^+,y^-} \Bigg[\sum_{i=1}^{p} \exp\left(-\lambda_i z^2\right) \left(-\frac{2\lambda_i z}{\sigma^2} \phi_i(z) + \phi_i'(z)\right) \left(\nabla_\theta f_\theta(x, y^+) - \nabla_\theta f_\theta(x, y^-)\right) \\ + \gamma \sum_{i=1}^{p} \exp\left(-\lambda_i r^2\right) \left(-\frac{2\lambda_i r}{\sigma^2} \phi_i(r) + \phi_i'(r)\right) \\ \times \left(\frac{(e_x^\top e_{y^-})(\nabla_\theta e_x)^\top e_{y^+} + (e_x^\top e_{y^-}) e_x^\top (\nabla_\theta e_{y^+}) - (e_x^\top e_{y^+})(\nabla_\theta e_x)^\top e_{y^-} - (e_x^\top e_{y^+}) e_x^\top (\nabla_\theta e_{y^-})}{(e_x^\top e_{y^-})^2}\right)\Bigg]\end{aligned}$$

Interpretation of the Gradient

- **Log Probability Ratio Term**:

– $-\frac{2\lambda_i z}{\sigma^2}\phi_i(z)$: Scales the influence of the log probability ratio based on its magnitude, the spectral kernel parameter λ_i, and the bandwidth parameter σ.

– $\phi_i'(z)$: Incorporates the derivative of the feature transformation function, allowing for more nuanced adjustments based on the transformed log probability ratio.

– $\nabla_\theta z = \nabla_\theta \log \pi(y^+ \mid x) - \nabla_\theta \log \pi(y^- \mid x)$: Encourages the model to increase the probability of the positive sample y^+ and decrease the probability of the negative sample y^-.

- **Spectral Kernel Term**:

– $-\frac{2\lambda_i r}{\sigma^2}\phi_i(r)$: Scales the influence of the embedding-based ratio based on its magnitude, the spectral kernel parameter λ_i, and the bandwidth parameter σ.

– $\phi_i'(r)$: Incorporates the derivative of the feature transformation function, allowing for more nuanced adjustments based on the transformed embedding ratio.

– $\nabla_\theta r = \frac{(e_x^\top e_{y^-})\nabla_\theta(e_x^\top e_{y^+}) - (e_x^\top e_{y^+})\nabla_\theta(e_x^\top e_{y^-})}{(e_x^\top e_{y^-})^2}$: Adjusts the model to better align the embeddings of x with y^+ while discouraging alignment with y^-.

- **Hyperparameters**:

– λ_i: Controls the influence of each spectral kernel component.

– γ: Balances the influence between the log probability ratio term and the embedding-based term.

– σ: Determines the bandwidth of the RBF kernel, affecting how sharply the exponential terms decay.

I.14 Computational Complexity Analysis of Spectral Kernelized Hybrid Loss

To evaluate the efficiency of the Spectral Kernelized Hybrid Loss, we analyze the computational complexity of its two primary components: the log probability ratio term and the spectral kernel term.

1. Log Probability Ratio Term

The log probability ratio term is defined as:

$$\sum_{i=1}^{p} \exp\left(-\lambda_i z^2\right)\phi_i(z),$$

where $z = \log \frac{\pi(y^+|x)}{\pi(y^-|x)}$.

Steps Involved:

- **Score Computation**: Calculate $f_\theta(x, y)$ for each class y, which involves a dot product between input features and model parameters.
- **Softmax Calculation**: Compute the exponential $e^{f_\theta(x,y)}$ for each class and normalize by the sum over all C classes.
- **Log Probability Ratio**: Compute the logarithm of the ratio between the probabilities of the positive and negative classes.
- **Exponentiation and Feature Transformation**: For each spectral kernel component i, compute the exponential and apply the feature transformation function $\phi_i(z)$.

Time Complexity: $O(p \cdot C)$, where p is the number of spectral kernel components and C is the number of classes. This complexity arises from iterating over each spectral kernel component and performing computations that scale with C.

2. Spectral Kernel Term

The spectral kernel term is defined as:

$$\gamma \sum_{i=1}^{p} \exp\left(-\lambda_i r^2\right)\phi_i(r),$$

where $r = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.

Steps Involved:

- **Dot Product Computation**: Calculate the dot products $e_x^\top e_{y^+}$ and $e_x^\top e_{y^-}$, where $e_x, e_{y^+}, e_{y^-} \in \mathbb{R}^d$.
- **Ratio Calculation**: Compute the ratio $\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.
- **Exponentiation and Feature Transformation**: For each spectral kernel component i, compute the exponential and apply the feature transformation function $\phi_i(r)$.

Time Complexity: $O(p \cdot d)$, where p is the number of spectral kernel components and d is the dimension of the embeddings. This arises from iterating over each spectral kernel component and performing computations that scale with d.

Overall Computational Complexity

Combining both components, the total computational complexity of the **Spectral Kernelized Hybrid Loss** is:

$$O(p \cdot C + p \cdot d) = O(p(C + d)),$$

where:

- p is the number of spectral kernel components.
- C is the number of classes.
- d is the embedding dimension.

This linear complexity in p, C, and d ensures scalability for large-scale applications involving multiple spectral kernel components, high-dimensional embeddings, and extensive class labels.

Comparison with Standard Loss Functions

- **Cross-Entropy Loss**:

– **Time Complexity**: $O(C)$.

– **Description**: Involves computing the softmax over C classes and calculating the negative log-likelihood.

- **Contrastive Loss**:

– **Time Complexity**: $O(d)$.

– **Description**: Focuses on the distance between embeddings, typically requiring computation of pairwise distances.

- **Spectral Kernelized Hybrid Loss**:

– **Time Complexity**: $O(p(C + d))$.

– **Description**: Extends both Cross-Entropy and Contrastive Losses by incorporating multiple spectral kernel components, enhancing the model's ability to capture complex relationships while maintaining computational efficiency.

The **Spectral Kernelized Hybrid Loss** thus offers a comprehensive approach by integrating multiple spectral kernels into the loss function, providing enhanced modeling capabilities at a manageable computational cost.

I.15 Efficiency of Spectral Kernelized Hybrid Loss

The **Spectral Kernelized Hybrid Loss** achieves a balanced trade-off between discriminative power and computational efficiency through the following mechanisms:

- **Scalability with Spectral Components**: By allowing multiple spectral kernel components (p), the loss function can capture a variety of complex patterns and relationships in the data without a disproportionate increase in computational cost.
- **Linear Scaling**: The loss scales linearly with the number of spectral kernel components p, the number of classes C, and the embedding dimension d, ensuring that it remains efficient even as these parameters grow.
- **Parallel Computation**: Both the log probability ratio term and the spectral kernel term involve operations that can be parallelized across spectral kernel components. Leveraging modern hardware accelerators, such as GPUs, can significantly speed up these computations.

- **Integrated Feature Transformations**: The use of feature transformation functions $\phi_i(\cdot)$ allows for sophisticated transformations of the log probability ratios and embedding ratios, enriching the model's learning capacity without incurring substantial additional computational overhead.

- **Hyperparameter Flexibility**: The hyperparameters λ_i, γ, and σ provide flexibility in controlling the influence of each spectral kernel component and the overall balance between probability-based and embedding-based terms. This allows for fine-tuning to achieve optimal performance without significant computational penalties.

I.16 Practical Considerations

While the theoretical complexity of the **Spectral Kernelized Hybrid Loss** is $O(p(C+d))$, several practical factors can influence its real-world performance:

- **GPU Parallelism**: Leveraging GPU parallelism can mitigate the linear scaling with p, C, and d, allowing for efficient computation even with large numbers of spectral kernel components, classes, and high-dimensional embeddings.

- **Optimized Implementations**: Utilizing optimized libraries (e.g., BLAS, cuDNN) for matrix operations and gradient computations can enhance performance, reducing the actual computation time.

- **Batch Sizing**: Selecting appropriate batch sizes can maximize hardware utilization. Larger batches may improve computational efficiency but require more memory, while smaller batches may be more memory-efficient but less computationally optimal.

- **Hyperparameter Tuning**: Careful tuning of the hyperparameters γ, λ_i, and σ is essential. Higher values of p can capture more complex relationships but may increase computational cost and risk overfitting. Similarly, the bandwidth parameter σ affects how sharply the exponential terms decay, influencing the gradient magnitudes.

- **Numerical Stability**: The constants c and σ ensure numerical stability, especially when dealing with small or large ratios in the log probability and embedding terms. Properly choosing these constants is crucial to prevent numerical issues during training.

- **Memory Consumption**: As p, C, and d increase, memory consumption can become a bottleneck. Efficient memory management and possibly reducing the number of spectral kernel components p without significantly compromising performance can help mitigate this issue.

By considering these practical aspects, the **Spectral Kernelized Hybrid Loss** can be effectively integrated into large-scale machine learning models, providing enhanced performance through sophisticated spectral kernel transformations while maintaining computational efficiency.

I.17 Gradient of Mahalanobis Kernelized Hybrid Loss

The **Mahalanobis Kernelized Hybrid Loss** is defined as:

$$\mathcal{L} = \mathbb{E}_{x,y^+,y^-}\left[\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)}-\mu\right)^2}{2\sigma^2}\right)+\gamma\exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}-\mu'\right)^2}{2\sigma'^2}\right)\right],$$

where:

- x represents the input data.

- y^+ and y^- denote the positive and negative samples, respectively.

- $\pi(y \mid x)$ is the probability of y given x, modeled using a softmax function.

- e_y and e_x are the embeddings of y and x, respectively.

- μ and μ' are means for the log probability ratio and embedding ratio terms, respectively.
- σ and σ' are bandwidth parameters for the Mahalanobis kernels applied to the log probability ratio and embedding ratio terms, respectively.
- γ is a hyperparameter controlling the influence of the embedding-based term.

Our objective is to compute the gradient of the Mahalanobis Kernelized Hybrid Loss $\nabla_\theta \mathcal{L}$ with respect to the model parameters θ. This involves differentiating each term of the loss function separately and then combining them.

Gradient of the Log Probability Ratio Term

The first component of the Mahalanobis Kernelized Hybrid Loss involves the exponential of the squared and shifted log probability ratio:

$$\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)} - \mu\right)^2}{2\sigma^2}\right).$$

To compute its gradient with respect to θ, we apply the chain rule:

$$\nabla_\theta \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) = \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)\cdot\left(-\frac{2(z-\mu)}{2\sigma^2}\right)\cdot\nabla_\theta z,$$

where $z = \log\frac{\pi(y^+|x)}{\pi(y^-|x)}$.

Simplifying, we obtain:

$$\nabla_\theta \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) = -\frac{(z-\mu)}{\sigma^2}\exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)\cdot\nabla_\theta z.$$

Expanding the gradient of the log probability ratio:

$$\nabla_\theta z = \nabla_\theta \log\frac{\pi(y^+ \mid x)}{\pi(y^- \mid x)} = \nabla_\theta \log \pi(y^+ \mid x) - \nabla_\theta \log \pi(y^- \mid x).$$

Assuming $\pi_\theta(y \mid x)$ is modeled using a softmax function:

$$\pi_\theta(y \mid x) = \frac{e^{f_\theta(x,y)}}{\sum_{y'} e^{f_\theta(x,y')}},$$

the gradient of $\log \pi(y \mid x)$ with respect to θ is:

$$\nabla_\theta \log \pi(y \mid x) = \nabla_\theta f_\theta(x,y) - \sum_{y'} \pi_\theta(y' \mid x)\nabla_\theta f_\theta(x,y').$$

Substituting back, we obtain:

$$\nabla_\theta z = \left[\nabla_\theta f_\theta(x,y^+) - \sum_{y'} \pi_\theta(y' \mid x)\nabla_\theta f_\theta(x,y')\right] - \left[\nabla_\theta f_\theta(x,y^-) - \sum_{y'} \pi_\theta(y' \mid x)\nabla_\theta f_\theta(x,y')\right]$$
$$= \nabla_\theta f_\theta(x,y^+) - \nabla_\theta f_\theta(x,y^-)$$

Therefore, the gradient of the log probability ratio term is:

$$\nabla_\theta \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) = -\frac{(z-\mu)}{\sigma^2}\exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)\left(\nabla_\theta f_\theta(x,y^+) - \nabla_\theta f_\theta(x,y^-)\right).$$

Gradient of the Mahalanobis Kernel Term

The second component involves the exponential of the squared and shifted embedding-based ratio:

$$\gamma\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right),$$

where $r = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.

To compute its gradient with respect to θ, we apply the chain rule:

$$\nabla_\theta \gamma\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right) = \gamma\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right)\cdot\left(-\frac{2(r-\mu')}{2\sigma'^2}\right)\cdot\nabla_\theta r,$$

where $r = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.

Simplifying, we obtain:

$$\nabla_\theta \gamma\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right) = -\frac{\gamma(r-\mu')}{\sigma'^2}\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right)\cdot\nabla_\theta r.$$

To compute $\nabla_\theta r$, we use the quotient rule:

$$\nabla_\theta r = \nabla_\theta\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}\right) = \frac{(e_x^\top e_{y^-})\nabla_\theta(e_x^\top e_{y^+}) - (e_x^\top e_{y^+})\nabla_\theta(e_x^\top e_{y^-})}{(e_x^\top e_{y^-})^2}.$$

Assuming e_x and e_y are differentiable with respect to θ, we have:

$$\nabla_\theta(e_x^\top e_y) = (\nabla_\theta e_x)^\top e_y + e_x^\top(\nabla_\theta e_y).$$

Substituting back, the gradient of the Mahalanobis kernel term becomes:

$$\nabla_\theta \gamma\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right) = -\frac{\gamma(r-\mu')}{\sigma'^2}\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right)\cdot\frac{(e_x^\top e_{y^-})\nabla_\theta(e_x^\top e_{y^+}) - (e_x^\top e_{y^+})\nabla_\theta(e_x^\top e_{y^-})}{(e_x^\top e_{y^-})^2}$$

Combined Gradient

Combining the gradients of both components, the overall gradient of the Mahalanobis Kernelized Hybrid Loss with respect to θ is:

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{x,y^+,y^-}\Bigg[-\frac{(z-\mu)}{\sigma^2}\exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)\left(\nabla_\theta f_\theta(x,y^+) - \nabla_\theta f_\theta(x,y^-)\right)$$

$$-\frac{\gamma(r-\mu')}{\sigma'^2}\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right)\cdot\frac{(e_x^\top e_{y^-})\nabla_\theta(e_x^\top e_{y^+}) - (e_x^\top e_{y^+})\nabla_\theta(e_x^\top e_{y^-})}{(e_x^\top e_{y^-})^2}\Bigg].$$

Simplified Gradient Expression For ease of implementation and readability, the gradient can be expressed as:

$$\nabla_\theta \mathcal{L} = \mathbb{E}_{x,y^+,y^-}\Bigg[-\frac{(z-\mu)}{\sigma^2}\exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)\left(\nabla_\theta f_\theta(x,y^+) - \nabla_\theta f_\theta(x,y^-)\right)$$

$$-\frac{\gamma(r-\mu')}{\sigma'^2}\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right)\cdot\frac{(e_x^\top e_{y^-})\left((\nabla_\theta e_x)^\top e_{y^+} + e_x^\top(\nabla_\theta e_{y^+})\right) - (e_x^\top e_{y^+})\left((\nabla_\theta e_x)^\top e_{y^-} + e_x^\top(\nabla_\theta e_{y^-})\right)}{(e_x^\top e_{y^-})^2}\Bigg].$$

Interpretation of the Gradient

- **Log Probability Ratio Term**:

– $-\frac{(z-\mu)}{\sigma^2}\exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$: Scales the influence of the log probability ratio based on its deviation from the mean μ and the bandwidth parameter σ.

– $\nabla_\theta f_\theta(x,y^+)$: Encourages the model to increase the score (and hence the probability) of the positive sample y^+.

– $-\nabla_\theta f_\theta(x,y^-)$: Encourages the model to decrease the score (and hence the probability) of the negative sample y^-.

- **Mahalanobis Kernel Term**:

– $-\frac{\gamma(r-\mu')}{\sigma'^2}\exp\left(-\frac{(r-\mu')^2}{2\sigma'^2}\right)$: Scales the influence of the embedding-based ratio based on its deviation from the mean μ' and the bandwidth parameter σ', adjusted by the hyperparameter γ.

– $\frac{(e_x^\top e_{y-})\nabla_\theta(e_x^\top e_{y+}) - (e_x^\top e_{y+})\nabla_\theta(e_x^\top e_{y-})}{(e_x^\top e_{y-})^2}$: Adjusts the model to better align the embeddings of x with y^+ while discouraging alignment with y^-.

– The exponential term $\exp\left(-\frac{(r-\mu')^2}{2\,(\sigma')^2}\right)$ ensures that the influence diminishes as the squared ratios deviate from the mean μ', promoting smoother gradients.

- **Hyperparameters**:

– μ and μ': Control the center of the Mahalanobis kernels for the log probability ratio and embedding ratio terms, respectively.

– σ and σ': Determine the bandwidth of the Mahalanobis kernels, affecting how sharply the exponential terms decay.

– γ: Balances the influence between the probability-based term and the embedding-based term, allowing for fine-tuning of their relative importance.

I.18 Computational Complexity Analysis of Mahalanobis Kernelized Hybrid Loss

To evaluate the efficiency of the Mahalanobis Kernelized Hybrid Loss, we analyze the computational complexity of its two primary components: the log probability ratio term and the Mahalanobis kernel term.

1. Log Probability Ratio Term

The log probability ratio term is defined as:

$$\exp\left(-\frac{\left(\log\frac{\pi(y^+|x)}{\pi(y^-|x)} - \mu\right)^2}{2\sigma^2}\right)$$

where $\pi_\theta(y \mid x)$ is modeled using a softmax function:

$$\pi_\theta(y \mid x) = \frac{e^{f_\theta(x,y)}}{\sum_{y'} e^{f_\theta(x,y')}},$$

and $z = \log\frac{\pi(y^+|x)}{\pi(y^-|x)}$.

Steps Involved:

- **Score Computation**: Calculate $f_\theta(x,y)$ for each class y, which involves a dot product between input features and model parameters.

- **Softmax Calculation**: Compute the exponential $e^{f_\theta(x,y)}$ for each class and normalize by the sum over all C classes.

- **Log Probability Ratio**: Compute the logarithm of the ratio between the probabilities of the positive and negative classes.

- **Exponentiation and Scaling**: Subtract the mean μ, square the result, scale by $-\frac{1}{2\sigma^2}$, and compute the exponential.

Time Complexity: $O(C)$, where C is the number of classes. This complexity arises from the softmax computation, which requires evaluating $f_\theta(x, y)$ and normalizing over all C classes.

2. Mahalanobis Kernel Term

The Mahalanobis kernel term is defined as:

$$\gamma \exp\left(-\frac{\left(\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}} - \mu'\right)^2}{2\sigma'^2}\right)$$

where $r = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.

Steps Involved:

- **Dot Product Computation**: Calculate the dot products $e_x^\top e_{y^+}$ and $e_x^\top e_{y^-}$, where $e_x, e_{y^+}, e_{y^-} \in \mathbb{R}^d$.
- **Ratio Calculation**: Compute the ratio $\frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.
- **Mean Subtraction and Squaring**: Subtract the mean μ', square the result, and scale by $-\frac{1}{2\sigma'^2}$.
- **Exponentiation and Scaling**: Compute the exponential and multiply by the hyperparameter γ.

Time Complexity: $O(d)$, where d is the dimension of the embeddings. This arises from the computation of the dot products between e_x and e_y, which scales linearly with d.

Overall Computational Complexity

Combining both components, the total computational complexity of the **Mahalanobis Kernelized Hybrid Loss** is:

$$O(C) + O(d) = O(C + d),$$

where:

- C is the number of classes (softmax computation).
- d is the embedding dimension (kernel computation).

This linear complexity ensures scalability for large-scale applications involving high-dimensional embeddings and extensive class labels.

Comparison with Standard Loss Functions

- **Cross-Entropy Loss**:
  - **Time Complexity**: $O(C)$.
  - **Description**: Involves computing the softmax over C classes and calculating the negative log-likelihood.
- **Contrastive Loss**:
  - **Time Complexity**: $O(d)$.
  - **Description**: Focuses on the distance between embeddings, typically requiring computation of pairwise distances.
- **Mahalanobis Kernelized Hybrid Loss**:
  - **Time Complexity**: $O(C + d)$.
  - **Description**: Combines both the discriminative power of the log probability ratio (similar to Cross-Entropy Loss) and the semantic richness of the Mahalanobis kernel (similar to Contrastive Loss), thereby integrating both aspects into a single loss function.

The **Mahalanobis Kernelized Hybrid Loss** thus offers a balanced combination of the computational efficiencies of Cross-Entropy and Contrastive Losses while enhancing the model's ability to capture both discriminative and semantic relationships.

I.19 Efficiency of Mahalanobis Kernelized Hybrid Loss

The **Mahalanobis Kernelized Hybrid Loss** achieves a balanced trade-off between discriminative power and computational efficiency through the following mechanisms:

- **Linear Scaling**: The loss scales linearly with both the number of classes C and the embedding dimension d, ensuring scalability for large-scale datasets and high-dimensional embedding spaces.
- **Parallel Computation**: Both the log probability ratio term and the Mahalanobis kernel term can be computed in parallel. Modern hardware accelerators, such as GPUs, can leverage this parallelism to significantly speed up training processes.
- **Integrated Semantic Information**: By combining probability-based and embedding-based objectives, the loss function enriches the model's learning without incurring substantial additional computational overhead.
- **Hyperparameter Control**: The hyperparameters μ, μ', σ, σ', and γ allow for fine-tuning the influence of each component, enabling the model to balance between accurately classifying positive and negative samples and capturing meaningful embedding relationships.

I.20 Practical Considerations

While the theoretical complexity of the Mahalanobis Kernelized Hybrid Loss is $O(C + d)$, several practical factors can influence its real-world performance:

- **GPU Parallelism**: Leveraging GPU parallelism can mitigate the linear scaling with C and d, allowing for efficient computation even with large numbers of classes and high-dimensional embeddings.
- **Optimized Implementations**: Utilizing optimized libraries (e.g., BLAS, cuDNN) for matrix operations and gradient computations can enhance performance, reducing the actual computation time.
- **Batch Sizing**: Selecting appropriate batch sizes can maximize hardware utilization. Larger batches may improve computational efficiency but require more memory, while smaller batches may be more memory-efficient but less computationally optimal.
- **Hyperparameter Tuning**: Careful tuning of the hyperparameters μ, μ', σ, σ', and γ is essential. The means μ and μ' determine the centers of the Mahalanobis kernels, while the bandwidth parameters σ and σ' affect how sharply the exponential terms decay. The hyperparameter γ balances the influence between the probability-based and embedding-based terms.
- **Numerical Stability**: The constants μ, μ', σ, and σ' ensure numerical stability, especially when dealing with small or large ratios in the log probability and embedding terms. Properly choosing these constants is crucial to prevent numerical issues during training.
- **Memory Consumption**: As C and d increase, memory consumption can become a bottleneck. Efficient memory management and possibly reducing the number of classes or embedding dimensions without significantly compromising performance can help mitigate this issue.

By considering these practical aspects, the **Mahalanobis Kernelized Hybrid Loss** can be effectively integrated into large-scale machine learning models, providing enhanced performance through sophisticated kernel transformations while maintaining computational efficiency.

I.21 Gradient of Hierarchical Mixture of Kernels (HMK)

Hierarchical Mixture of Kernels (HMK) imposes a hierarchical structure where local kernels operate on small, local regions, and global kernels capture larger-scale dependencies. This structure is formalized as:

$$\begin{aligned}K(x,x') = \tau_1 \left(\lambda_1 K_{\text{RBF}}(x,x') + \lambda_2 K_{\text{Poly}}(x,x')\right) \\ + \tau_2 \left(\lambda_3 K_{\text{Spectral}}(x,x') + \lambda_4 K_{\text{Mahalanobis}}(x,x')\right)\end{aligned}$$

where:

- x, x' are input data points.
- $K_{\text{RBF}}(x,x')$, $K_{\text{Poly}}(x,x')$, $K_{\text{Spectral}}(x,x')$, and $K_{\text{Mahalanobis}}(x,x')$ are the Radial Basis Function, Polynomial, Spectral, and Mahalanobis kernels, respectively.
- $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weighting coefficients for each kernel type.
- τ_1 and τ_2 are scaling factors that balance the contribution of local and global kernels.

Our objective is to compute the gradient of the HMK $K(x,x')$ with respect to the model parameters θ. This gradient is essential for optimizing the model parameters during training, ensuring that both local and global dependencies are appropriately captured.

Gradient Computation of HMK

The gradient of the HMK with respect to θ is derived by differentiating each component kernel individually and then combining them according to their hierarchical structure. Formally, the gradient is expressed as:

$$\nabla_\theta K(x,x') = \tau_1 \left(\lambda_1 \nabla_\theta K_{\text{RBF}}(x,x') + \lambda_2 \nabla_\theta K_{\text{Poly}}(x,x')\right)$$

$$+\tau_2 \left(\lambda_3 \nabla_\theta K_{\text{Spectral}}(x,x') + \lambda_4 \nabla_\theta K_{\text{Mahalanobis}}(x,x')\right)$$

Each gradient term $\nabla_\theta K_{\text{Type}}(x,x')$ corresponds to the gradient of the respective kernel with respect to θ, as derived in their individual sections.

1. Gradient of the RBF Kernel

$$\nabla_\theta K_{\text{RBF}}(x,x') = \nabla_\theta \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right) \cdot \left(\frac{(x'-x)}{\sigma^2}\right) \cdot \nabla_\theta x,$$

where σ is the bandwidth parameter.

2. Gradient of the Polynomial Kernel

$$\nabla_\theta K_{\text{Poly}}(x,x') = \nabla_\theta (x^\top x' + c)^d = d(x^\top x' + c)^{d-1} \cdot \left(x' \nabla_\theta x + x \nabla_\theta x'\right),$$

where c is a constant and d is the degree of the polynomial.

3. Gradient of the Spectral Kernel

$$\nabla_\theta K_{\text{Spectral}}(x,x') = \sum_{i=1}^{p} \left[\exp\left(-\lambda_i z_i^2\right)\left(-2\lambda_i z_i \phi_i(z_i) + \phi_i'(z_i)\right) \nabla_\theta z_i\right],$$

where $z_i = \log \frac{\pi(y^+|x)}{\pi(y^-|x)}$ and $\phi_i(\cdot)$ are feature transformation functions.

4. Gradient of the Mahalanobis Kernel

$$\nabla_\theta K_{\text{Mahalanobis}}(x,x') = \sum_{i=1}^{p} \left[\exp\left(-\lambda_i (r_i-\mu_i)^2\right)\left(-\frac{2\lambda_i(r_i-\mu_i)}{\sigma_i^2}\phi_i(r_i) + \phi_i'(r_i)\right) \nabla_\theta r_i\right],$$

where $r_i = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$, μ_i are mean parameters, and σ_i are bandwidth parameters for each spectral component.

Combined Gradient Expression

Combining the gradients of all kernel components, the overall gradient of the HMK with respect to θ is:

$$\nabla_\theta K(x,x') = \tau_1 \left(\lambda_1 \nabla_\theta K_{\text{RBF}}(x,x') + \lambda_2 \nabla_\theta K_{\text{Poly}}(x,x')\right)$$

$$+\tau_2 \left(\lambda_3 \nabla_\theta K_{\text{Spectral}}(x,x') + \lambda_4 \nabla_\theta K_{\text{Mahalanobis}}(x,x')\right).$$

Substituting the gradients of individual kernels:

$$\nabla_\theta K(x,x') = \tau_1 \Bigg(\lambda_1 \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right) \cdot \frac{(x'-x)}{\sigma^2} \cdot \nabla_\theta x$$

$$+\lambda_2 d(x^\top x' + c)^{d-1} \cdot \left(x' \nabla_\theta x + x \nabla_\theta x'\right)\Bigg)$$

$$+\tau_2 \Bigg(\lambda_3 \sum_{i=1}^{p} \exp\left(-\lambda_i z_i^2\right)\left(-2\lambda_i z_i \phi_i(z_i) + \phi_i'(z_i)\right) \nabla_\theta z_i$$

$$+\lambda_4 \sum_{i=1}^{p} \exp\left(-\lambda_i (r_i-\mu_i)^2\right)\left(-\frac{2\lambda_i(r_i-\mu_i)}{\sigma_i^2}\phi_i(r_i) + \phi_i'(r_i)\right) \nabla_\theta r_i\Bigg).$$

Simplified Gradient Expression For ease of implementation and readability, the gradient can be succinctly written as:

$$\nabla_\theta K(x,x') = \tau_1 \lambda_1 \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right) \cdot \frac{(x'-x)}{\sigma^2} \cdot \nabla_\theta x$$

$$+\tau_1 \lambda_2 d(x^\top x' + c)^{d-1} \cdot \left(x' \nabla_\theta x + x \nabla_\theta x'\right)$$

$$+\tau_2\lambda_3 \sum_{i=1}^{p} \exp\left(-\lambda_i z_i^2\right)\left(-2\lambda_i z_i \phi_i(z_i) + \phi_i'(z_i)\right)\nabla_\theta z_i$$

$$+\tau_2\lambda_4 \sum_{i=1}^{p} \exp\left(-\lambda_i (r_i-\mu_i)^2\right)\left(-\frac{2\lambda_i(r_i-\mu_i)}{\sigma_i^2}\phi_i(r_i) + \phi_i'(r_i)\right)\nabla_\theta r_i.$$

Interpretation of the Gradient

- **RBF Kernel Gradient ($\tau_1\lambda_1$):**
  - $\exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$: Measures the similarity between x and x'.
  - $\frac{(x'-x)}{\sigma^2}$: Directs the gradient to increase similarity if x and x' are similar, or decrease otherwise.
  - $\nabla_\theta x$: Adjusts the model parameters to optimize the representation of x.
- **Polynomial Kernel Gradient ($\tau_1\lambda_2$):**
  - $d(x^\top x' + c)^{d-1}$: Scales the influence based on the degree of the polynomial and the similarity between x and x'.
  - $(x'\nabla_\theta x + x\nabla_\theta x')$: Updates the model parameters to enhance or reduce the polynomial similarity.
- **Spectral Kernel Gradient ($\tau_2\lambda_3$):**
  - $\exp\left(-\lambda_i z_i^2\right)$: Applies a spectral transformation based on the log probability ratio.
  - $(-2\lambda_i z_i \phi_i(z_i) + \phi_i'(z_i))$: Modulates the gradient based on the spectral feature transformations.
  - $\nabla_\theta z_i$: Encourages the model to adjust probabilities to optimize the spectral features.
- **Mahalanobis Kernel Gradient ($\tau_2\lambda_4$):**
  - $\exp\left(-\lambda_i(r_i-\mu_i)^2\right)$: Applies a Mahalanobis transformation based on the embedding ratio.
  - $\left(-\frac{2\lambda_i(r_i-\mu_i)}{\sigma_i^2}\phi_i(r_i) + \phi_i'(r_i)\right)$: Modulates the gradient based on the Mahalanobis feature transformations.
  - $\nabla_\theta r_i$: Adjusts the embeddings to optimize the Mahalanobis distance.
- **Hyperparameters $\tau_1, \tau_2, \lambda_1, \lambda_2, \lambda_3, \lambda_4$:**
  - τ_1, τ_2: Balance the contributions of local and global kernels.
  - $\lambda_1, \lambda_2, \lambda_3, \lambda_4$: Control the influence of each kernel type within their respective hierarchies.

Computational Complexity Analysis of HMK

To evaluate the efficiency of the Hierarchical Mixture of Kernels (HMK), we analyze the computational complexity of its primary components: the local kernels (RBF and Polynomial) and the global kernels (Spectral and Mahalanobis).

1. Local Kernels

a. RBF Kernel

$$K_{\text{RBF}}(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$$

Steps Involved:

- Compute the Euclidean distance $\|x-x'\|$, which involves $O(d)$ operations, where d is the dimension of the input.
- Exponentiation, which is a constant-time operation.

Time Complexity: $O(d)$

b. Polynomial Kernel

$$K_{\text{Poly}}(x,x') = (x^\top x' + c)^d$$

Steps Involved:

- Compute the dot product $x^\top x'$, which involves $O(d)$ operations.
- Add constant c and raise to the power d, both of which are constant-time operations.

Time Complexity: $O(d)$

2. Global Kernels

a. Spectral Kernel

$$K_{\text{Spectral}}(x, x') = \sum_{i=1}^{p} \exp\left(-\lambda_i z_i^2\right) \phi_i(z_i),$$

where $z_i = \log \frac{\pi(y^+|x)}{\pi(y^-|x)}$.

Steps Involved:

- Compute the log probability ratio z_i, which involves $O(C)$ operations due to the softmax.
- For each of the p spectral components:

– Compute $\exp\left(-\lambda_i z_i^2\right)$, which is a constant-time operation.

– Apply the feature transformation $\phi_i(z_i)$, assumed to be constant-time.

Time Complexity: $O(p \cdot C)$

b. Mahalanobis Kernel

$$K_{\text{Mahalanobis}}(x, x') = \sum_{i=1}^{p} \exp\left(-\lambda_i (r_i - \mu_i)^2\right) \phi_i(r_i),$$

where $r_i = \frac{e_x^\top e_{y^+}}{e_x^\top e_{y^-}}$.

Steps Involved:

- Compute the embedding ratios r_i, which involves $O(d)$ operations for the dot products.
- For each of the p Mahalanobis components:

– Compute $\exp\left(-\lambda_i (r_i - \mu_i)^2\right)$, which is a constant-time operation.

– Apply the feature transformation $\phi_i(r_i)$, assumed to be constant-time.

Time Complexity: $O(p \cdot d)$

Overall Computational Complexity

Combining the complexities of all kernel components, the total computational complexity of the **HMK** is:

$$O(d) + O(d) + O(p \cdot C) + O(p \cdot d) = O(p \cdot (C + d) + d).$$

Since d is typically much smaller than $p \cdot (C + d)$, the dominant term is $O(p \cdot (C + d))$.

Comparison with Standard Loss Functions

- **Cross-Entropy Loss**:

– **Time Complexity**: $O(C)$.

– **Description**: Involves computing the softmax over C classes and calculating the negative log-likelihood.

- **Contrastive Loss**:

– **Time Complexity**: $O(d)$.

– **Description**: Focuses on the distance between embeddings, typically requiring computation of pairwise distances.

- **Hierarchical Mixture of Kernels (HMK)**:

– **Time Complexity**: $O(p \cdot (C + d))$.

– **Description**: Combines multiple kernels with hierarchical weighting, integrating both local (RBF and Polynomial) and global (Spectral and Mahalanobis) dependencies. This allows HMK to capture complex patterns and relationships in the data, leveraging the strengths of each kernel type.

The **HMK** offers a more expressive and flexible modeling approach compared to standard loss functions by incorporating multiple kernel types and hierarchical weighting. However, this expressiveness comes at the cost of increased computational complexity, especially with higher numbers of spectral components p, classes C, and embedding dimensions d.

Efficiency of HMK

The **Hierarchical Mixture of Kernels (HMK)** achieves a balanced trade-off between modeling complexity and computational efficiency through the following mechanisms:

- **Modular Kernel Design**: By decomposing the kernel into local and global components, HMK allows for targeted optimization of different aspects of the data. Local kernels (RBF and Polynomial) focus on fine-grained similarities, while global kernels (Spectral and Mahalanobis) capture broader dependencies.

- **Parallel Computation**: The computations for different kernel components are independent and can be parallelized. Leveraging modern hardware accelerators, such as GPUs, can significantly reduce training times.
- **Scalability with Spectral Components**: Although the complexity scales with the number of spectral components p, careful selection of p can balance expressiveness with computational cost. Techniques such as dimensionality reduction or kernel approximation can be employed to manage large p.
- **Hyperparameter Tuning**: The hyperparameters $\tau_1, \tau_2, \lambda_1, \lambda_2, \lambda_3, \lambda_4$ allow for fine-tuning the influence of each kernel component, enabling the model to prioritize certain relationships over others without requiring extensive computational resources.
- **Optimized Implementations**: Utilizing optimized libraries (e.g., BLAS, cuDNN) for matrix operations and kernel computations can enhance performance, ensuring that the theoretical computational complexities translate into practical efficiency gains.

Practical Considerations

While the theoretical complexity of the **HMK** is $O(p \cdot (C + d))$, several practical factors can influence its real-world performance:

- **GPU Parallelism**: Leveraging GPU parallelism can mitigate the linear scaling with p, C, and d, allowing for efficient computation even with large numbers of spectral kernel components, classes, and high-dimensional embeddings.
- **Optimized Implementations**: Utilizing optimized libraries (e.g., BLAS, cuDNN) for matrix operations and gradient computations can enhance performance, reducing the actual computation time.
- **Batch Sizing**: Selecting appropriate batch sizes can maximize hardware utilization. Larger batches may improve computational efficiency but require more memory, while smaller batches may be more memory-efficient but less computationally optimal.
- **Hyperparameter Tuning**: Careful tuning of the hyperparameters $\tau_1, \tau_2, \lambda_1, \lambda_2, \lambda_3, \lambda_4$ is essential. The values of these parameters determine the relative importance of each kernel component, affecting both the model's performance and computational cost.
- **Memory Consumption**: As the number of spectral components p, classes C, and embedding dimensions d increase, memory consumption can become a bottleneck. Efficient memory management strategies, such as gradient checkpointing or dimensionality reduction, can help mitigate this issue.
- **Numerical Stability**: Ensuring numerical stability during kernel computations is crucial, especially when dealing with exponential functions that can lead to very large or very small values. Techniques such as normalization or adding small constants to denominators can prevent numerical overflow or underflow.
- **Kernel Selection**: The choice of kernel types and their respective parameters (σ, c, d, λ_i, μ_i, σ'_i) should be informed by the specific characteristics of the data and the problem domain. Empirical validation and cross-validation can aid in selecting optimal kernel configurations.

By considering these practical aspects, the **Hierarchical Mixture of Kernels (HMK)** can be effectively integrated into large-scale machine learning models, providing enhanced performance through sophisticated kernel combinations while maintaining computational efficiency.

I.22 Analysis of Gradient Convergence for Four Kernels and HMK

In this section, we investigate the convergence behavior of gradient descent when applied to four distinct kernels: **Polynomial**, **RBF (Radial Basis Function)**, **Spectral**, and **Mahalanobis**. Additionally, we analyze the **Hierarchical Mixture of Kernels (HMK)**, which combines these kernels to leverage both local and global dependencies. The convergence properties are evaluated based on key factors such as smoothness, Lipschitz continuity, gradient simplicity, and robustness to initialization. Understanding these properties is crucial for effective alignment learning and optimization.

I.23 Lipschitz Continuity: Intuition and Importance

Definition: A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be **Lipschitz continuous** with constant $L > 0$ if, for all $x, y \in \mathbb{R}^n$,

$$|f(x) - f(y)| \leq L\|x - y\|.$$

Here, L is the **Lipschitz constant** and serves as an upper bound on the rate at which the function f can change. Intuitively, Lipschitz continuity ensures that the function does not exhibit abrupt changes, which is essential for the stability and convergence of gradient-based optimization methods.

Why Lipschitz Continuity Matters

- **Convergence Stability**: If the gradient of a loss function is Lipschitz continuous, gradient descent is guaranteed to converge at a stable rate (Nesterov, 2003).
- **Prevention of Exploding Gradients**: Lipschitz continuity bounds the gradients, preventing excessively large updates that can destabilize the optimization process, particularly in deep learning models (Goodfellow et al., 2016).
- **Smooth Optimization Landscape**: A Lipschitz continuous gradient implies a smooth loss landscape, facilitating efficient and predictable optimization (Boyd and Vandenberghe, 2004).

Illustrative Examples

- **Lipschitz Continuous Function**: The linear function $f(x) = 2x$ is Lipschitz continuous with $L = 2$. Regardless of the input difference, the function's rate of change remains constant, ensuring bounded gradient updates.
- **Non-Lipschitz Function**: The quadratic function $f(x) = x^2$ is not Lipschitz continuous on $[0, \infty)$ because its slope becomes unbounded as x increases. This can lead to unstable gradient updates during optimization.

Relevance in Kernel Methods

The convergence behavior of different kernels is influenced by whether their gradients are Lipschitz continuous. Below, we explore how Lipschitz continuity impacts gradient descent for each of the four kernels under consideration.

I.24 Key Factors Influencing Gradient Convergence

To analyze the convergence of gradient descent for each kernel, we consider the following criteria:

- **Smoothness of the Loss Landscape**: A smoother loss landscape facilitates faster and more stable convergence by avoiding abrupt changes in gradients.
- **Lipschitz Continuity of the Gradient**: A smaller Lipschitz constant ensures that gradients do not change abruptly, promoting stable convergence (Nesterov, 2003).
- **Gradient Simplicity**: Simpler gradient expressions enhance computational efficiency and accelerate convergence.
- **Robustness to Initialization**: Kernels that exhibit less sensitivity to initial parameter values lead to more reliable convergence from diverse starting points.

I.25 Convergence Properties of Each Kernel

1. RBF Kernel

- **Smoothness**: The RBF kernel induces a smooth and convex loss landscape, which is conducive to fast and stable convergence (Bishop, 2006).

- **Lipschitz Continuity**: The gradient of the RBF kernel is Lipschitz continuous due to its exponential decay property. This ensures that gradient updates change gradually, enhancing convergence stability.

- **Gradient Simplicity**: The gradient of the RBF kernel is straightforward and linear with respect to the input:

  $$\nabla_y K_{\text{RBF}}(y, y') = K_{\text{RBF}}(y, y') \cdot \frac{(y' - y)}{\sigma^2},$$

  where σ is the bandwidth parameter.

- **Robustness to Initialization**: Due to its convex loss surface, the RBF kernel is robust to random initializations, minimizing the risk of converging to poor local minima (Schölkopf and Smola, 2002).

2. Polynomial Kernel

- **Smoothness**: The smoothness of the Polynomial kernel depends on its degree d. Higher degrees introduce non-convexity, resulting in a more rugged loss landscape with multiple local minima and saddle points.

- **Lipschitz Continuity**: Lipschitz continuity deteriorates as the degree d increases. Higher degrees lead to steeper gradients, making the optimization process more susceptible to instability.

- **Gradient Simplicity**: The gradient complexity increases with the degree d:

  $$\nabla_y K_{\text{Poly}}(y, y') = d(y^\top y' + c)^{d-1} \cdot y',$$

  where c is a constant.

- **Robustness to Initialization**: The Polynomial kernel is highly sensitive to initialization, especially for higher degrees, due to its non-convex loss landscape. This can lead to convergence to suboptimal local minima.

3. Spectral Kernel

- **Smoothness**: The smoothness of the Spectral kernel is influenced by the choice of basis functions ϕ_i. Orthonormal basis functions, such as wavelets, can introduce oscillatory behavior in the loss landscape (Ng et al., 2001).

- **Lipschitz Continuity**: Lipschitz continuity is contingent on the eigenvalues λ_i of the underlying Laplacian. Large eigenvalues can cause rapid oscillations in the gradients, leading to abrupt changes and potential instability.

- **Gradient Simplicity**: The gradient of the Spectral kernel depends on the complexity of the basis functions:

  $$\nabla_y K_{\text{Spectral}}(y, y') = \sum_{i=1}^{p} \left[\exp\left(-\lambda_i z_i^2\right)\left(-2\lambda_i z_i \phi_i(z_i) + \phi_i'(z_i)\right)\nabla_y z_i\right],$$

  where $z_i = \log \frac{\pi(y^+|x)}{\pi(y^-|x)}$.

- **Robustness to Initialization**: The convergence of the Spectral kernel is sensitive to the alignment between data and the chosen basis functions. Poor alignment can lead to oscillatory gradients, requiring careful initialization strategies (Ng et al., 2001).

4. Mahalanobis Kernel

- **Smoothness**: The Mahalanobis kernel behaves similarly to the RBF kernel when the covariance matrix Σ is the identity matrix. If Σ is poorly conditioned, the loss landscape becomes anisotropic, leading to uneven smoothness across different dimensions (Weinberger and Saul, 2009).

- **Lipschitz Continuity**: The Lipschitz continuity of the Mahalanobis kernel depends on the condition number of Σ. A well-conditioned Σ ensures smooth and stable gradients, while a poorly conditioned Σ results in rapidly changing gradients in certain directions.

- **Gradient Simplicity**: The gradient of the Mahalanobis kernel incorporates the precision matrix Σ^{-1}:

  $$\nabla_y K_{\text{Mahalanobis}}(y, y') = K_{\text{Mahalanobis}}(y, y') \cdot \Sigma^{-1}(y' - y).$$

  This introduces additional complexity compared to the RBF kernel.

- **Robustness to Initialization**: When Σ is well-conditioned, the Mahalanobis kernel exhibits robust convergence properties similar to the RBF kernel. However, a poorly conditioned Σ can lead to slow convergence and sensitivity to initialization due to uneven gradient magnitudes.

I.26 Convergence Properties of HMK

Hierarchical Mixture of Kernels (HMK) The **Hierarchical Mixture of Kernels (HMK)** integrates the four aforementioned kernels into a hierarchical structure to capture both local and global dependencies. HMK is defined as:

$$K(x, x') = \tau_1 \left(\lambda_1 K_{\text{RBF}}(x, x') + \lambda_2 K_{\text{Poly}}(x, x')\right) + \tau_2 \left(\lambda_3 K_{\text{Spectral}}(x, x') + \lambda_4 K_{\text{Mahalanobis}}(x, x')\right),$$

where:

- $K_{\text{RBF}}(x, x')$, $K_{\text{Poly}}(x, x')$, $K_{\text{Spectral}}(x, x')$, and $K_{\text{Mahalanobis}}(x, x')$ are the respective kernel functions.

- $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weighting coefficients for each kernel type.

- τ_1 and τ_2 are scaling factors that balance the contribution of local and global kernels.

- **Smoothness**: HMK exhibits **piecewise smoothness** due to its hierarchical decomposition. The local kernels (**RBF** and **Polynomial**) contribute to fine-grained similarities, while the global kernels (**Spectral** and **Mahalanobis**) capture broader dependencies. This combination allows HMK to adaptively smooth different regions of the loss landscape.

- **Lipschitz Continuity**: The Lipschitz continuity of HMK is influenced by the individual Lipschitz properties of its component kernels. Since **RBF** and **Mahalanobis** kernels typically have Lipschitz continuous gradients (when Σ is well-conditioned), and **Spectral** kernels have medium Lipschitz continuity depending on eigenvalues, HMK inherits a balanced Lipschitz continuity. The **Polynomial** kernel's Lipschitz properties can be controlled through the degree d, allowing HMK to maintain overall stability.

- **Gradient Simplicity**: The gradient of HMK is a weighted sum of the gradients of its individual kernels:

  $$\nabla_\theta K(x, x') = \tau_1 \left(\lambda_1 \nabla_\theta K_{\text{RBF}}(x, x') + \lambda_2 \nabla_\theta K_{\text{Poly}}(x, x')\right) + \tau_2 \left(\lambda_3 \nabla_\theta K_{\text{Spectral}}(x, x') + \lambda_4 \nabla_\theta K_{\text{Mahalanobis}}(x, x')\right).$$

  This modularity allows HMK to balance the simplicity of **RBF** and **Mahalanobis** gradients with the complexity of **Polynomial** and **Spectral** gradients, ensuring manageable gradient expressions.

- **Robustness to Initialization**: HMK enhances robustness to initialization by leveraging the stable convergence properties of **RBF** and **Mahalanobis** kernels alongside the expressive power of **Polynomial** and **Spectral** kernels. The hierarchical weighting factors τ_1 and τ_2 allow HMK to dynamically adjust the influence of each kernel type, reducing sensitivity to poor initializations.

I.27 Summary of Convergence Properties

The analysis reveals that each kernel exhibits distinct convergence behaviors influenced by their inherent properties:

- **RBF Kernel**: Offers the smoothest and most stable convergence due to its convex and Lipschitz continuous gradient. Its simplicity in gradient computation and robustness to initialization make it highly reliable for gradient-based optimization.

- **Polynomial Kernel**: Suffers from non-convexity and increasing Lipschitz constants with higher degrees. The complexity of its gradients and sensitivity to initialization can hinder stable convergence, especially for large d.

- **Spectral Kernel**: Introduces oscillatory behavior depending on the basis functions and eigenvalues. While it can capture intricate patterns, the potential for abrupt gradient changes requires careful design and initialization to ensure stable convergence.

- **Mahalanobis Kernel**: Balances between the RBF and Spectral kernels. With a well-conditioned covariance matrix Σ, it maintains smooth and Lipschitz continuous gradients, ensuring robust convergence. However, a poorly conditioned Σ can compromise convergence stability.

- **Hierarchical Mixture of Kernels (HMK)**: Combines the strengths of all four kernels, achieving a balanced convergence behavior. HMK benefits from the smoothness and stability of **RBF** and **Mahalanobis** kernels while incorporating the expressive power of **Polynomial** and **Spectral** kernels. This hierarchical structure ensures that HMK can adapt to various data characteristics, promoting both robust and efficient convergence.

Key Takeaways

- **RBF Kernel**: Ideal for scenarios requiring stable and rapid convergence. Its convexity and smooth gradients make it a dependable choice for many optimization tasks.

- **Polynomial Kernel**: Best suited for problems where capturing high-degree interactions is essential. However, care must be taken to manage its non-convexity and gradient complexity, particularly with higher degrees.

- **Spectral Kernel**: Effective in capturing complex, oscillatory patterns within the data. Requires careful selection of basis functions and initialization strategies to maintain convergence stability.

- **Mahalanobis Kernel**: Provides flexibility in modeling by incorporating covariance structure. Ensuring that Σ is well-conditioned is crucial for maintaining smooth and stable convergence.

- **Hierarchical Mixture of Kernels (HMK)**: Combines the strengths of all four kernels, offering a balanced and robust convergence behavior. HMK's hierarchical structure allows it to adapt to various data complexities, ensuring both stability and expressiveness in gradient-based optimization.

Designing kernels with favorable convergence properties is essential for robust and efficient optimization in alignment learning. Selecting the appropriate kernel based on the specific requirements of the task and the nature of the data can significantly enhance the performance and reliability of gradient-based learning algorithms.

For practical alignment tasks, the choice of kernel should balance computational complexity, convergence speed, and robustness. Hybrid approaches, such as the Hierarchical Mixture of Kernels (HMK) (Bach et al., 2004), leverage the strengths of multiple kernels to achieve more stable and generalizable learning outcomes.

I.28 Analysis of Kernel Properties Across Divergence Measures

This subsection provides a comprehensive analysis of kernel properties across various divergence measures, including **Kullback–Leibler (KL)**, **Jensen–Shannon (JS)**, **Hellinger**, **Rényi Divergence**, **Bhattacharyya**, **Wasserstein**, and **f-Divergence**. The focus is on four widely used kernels: **RBF (Radial Basis Function)**, **Polynomial**, **Spectral**, and **Mahalanobis**. For each kernel and divergence measure, the following key aspects are evaluated:

- **Smoothness**: Characterizes the landscape of the loss surface induced by each kernel under the respective divergence measure.

Table 8: Comparison of Gradient Convergence Properties for Four Kernels and HMK

Kernel	Smoothness	Lipschitz Gradient	Gradient Simplicity	Robustness to Initialization
RBF	Smooth, Convex	High	Simple	Robust
Polynomial	Non-Convex (Higher d)	Low (Higher d)	Complex	Sensitive
Spectral	Oscillatory (Basis-Dependent)	Medium	Complex (Basis-Dependent)	Moderate
Mahalanobis	Smooth (if Σ Well-Conditioned)	High (if Σ Well-Conditioned)	Similar to RBF	Robust (if Σ Well-Conditioned)
HMK	Piecewise Smooth	High	Composite of Simple and Complex	Highly Robust

- **Lipschitz Continuity**: Assesses the smoothness of gradient changes, where higher Lipschitz continuity is desirable for stable gradient descent.

- **Gradient Simplicity**: Evaluates the complexity of the gradient function, impacting computation time and convergence speed.

- **Robustness to Initialization**: Measures the sensitivity of convergence to the initial weights or parameters.

Key Observations from Table 9:

- **RBF Kernel**: The RBF kernel exhibits the most stable properties across all divergence measures. It maintains a **smooth, convex** loss landscape, **high Lipschitz continuity**, and **simple linear gradients**. These features contribute to **robust convergence**, making it a preferred choice in practical applications.

- **Polynomial Kernel**: The Polynomial kernel's properties are highly sensitive to its degree d. For large d, it becomes **non-convex**, with sharp transitions in its gradient. This increases its susceptibility to **poor initialization** and slower convergence. Additionally, its complexity increases as the degree d increases.

- **Spectral Kernel**: The Spectral kernel's behavior is highly dependent on the choice of basis functions $\phi_i(y)$. For certain bases, such as wavelets, the loss landscape becomes **oscillatory**, and convergence depends on the alignment of the initialization with the basis functions.

- **Mahalanobis Kernel**: The Mahalanobis kernel behaves similarly to the RBF kernel when $\Sigma = I$. For well-conditioned Σ, its properties remain stable and akin to the RBF kernel. However, if Σ is ill-conditioned, the loss landscape becomes **anisotropic**, leading to convergence slowdowns in specific directions.

- **Hierarchical Mixture of Kernels (HMK)**: HMK combines the strengths of all four kernels, achieving a balanced convergence behavior. It benefits from the **smoothness** and **stability** of the RBF and Mahalanobis kernels while incorporating the **expressive power** of the Polynomial and Spectral kernels. This hierarchical structure allows HMK to adapt to various data characteristics, promoting both robust and efficient convergence.

Implications for Kernel Selection: The choice of kernel in alignment tasks significantly impacts the optimization process. The RBF kernel is ideal for scenarios requiring stable and rapid convergence due to its smooth and convex properties. In contrast, the Polynomial kernel is suitable for modeling complex, high-degree interactions but demands careful tuning to manage its non-convexity and gradient complexity. The Spectral kernel excels in capturing oscillatory patterns, making it well-suited for graph-based data, whereas the Mahalanobis kernel offers flexibility in modeling anisotropic similarities, provided that the covariance matrix Σ is well-conditioned.

The HMK stands out by integrating multiple kernels to balance their respective strengths and mitigate their weaknesses. This hierarchical approach ensures that the optimization process remains robust and efficient across diverse data distributions

Table 9: Comparison of Kernels across Divergence Measures: KL, JS, Hellinger, Rényi, Bhattacharyya, Wasserstein, and f-Divergence

Kernel	Kullback–Leibler (KL)	Jensen–Shannon (JS)	Hellinger	Rényi Divergence	Bhattacharyya	Wasserstein	f-Divergence
RBF	Smooth and Convex	Smooth and Convex	Smooth and Convex	Smooth and Convex	Smooth and Convex	Smooth and Convex	Smooth and Convex
	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity
	Simple, Linear Gradients	Simple, Linear Gradients	Simple, Linear Gradients	Simple, Linear Gradients	Simple, Linear Gradients	Simple, Linear Gradients	Simple, Linear Gradients
	Robust and Fast Convergence	Robust and Fast Convergence	Robust and Fast Convergence	Robust and Fast Convergence	Robust and Fast Convergence	Robust and Fast Convergence	Robust and Fast Convergence
Polynomial	Complex and Non-Convex	Complex and Non-Convex	Complex and Non-Convex	Complex and Non-Convex	Complex and Non-Convex	Complex and Non-Convex	Complex and Non-Convex
	Low Lipschitz Continuity (High d)	Low Lipschitz Continuity (High d)	Low Lipschitz Continuity (High d)	Low Lipschitz Continuity (High d)	Low Lipschitz Continuity (High d)	Low Lipschitz Continuity (High d)	Low Lipschitz Continuity (High d)
	Non-Linear Gradients	Non-Linear Gradients	Non-Linear Gradients	Non-Linear Gradients	Non-Linear Gradients	Non-Linear Gradients	Non-Linear Gradients
	Sensitive to Initialization	Sensitive to Initialization	Sensitive to Initialization	Sensitive to Initialization	Sensitive to Initialization	Sensitive to Initialization	Sensitive to Initialization
Spectral	Oscillatory	Oscillatory	Oscillatory	Oscillatory	Oscillatory	Oscillatory	Oscillatory
	Medium Lipschitz Continuity (Basis-Dependent)	Medium Lipschitz Continuity (Basis-Dependent)	Medium Lipschitz Continuity (Basis-Dependent)	Medium Lipschitz Continuity (Basis-Dependent)	Medium Lipschitz Continuity (Basis-Dependent)	Medium Lipschitz Continuity (Basis-Dependent)	Medium Lipschitz Continuity (Basis-Dependent)
	Complex Gradients	Complex Gradients	Complex Gradients	Complex Gradients	Complex Gradients	Complex Gradients	Complex Gradients
	Moderate Sensitivity	Moderate Sensitivity	Moderate Sensitivity	Moderate Sensitivity	Moderate Sensitivity	Moderate Sensitivity	Moderate Sensitivity
Mahalanobis	Smooth (like RBF)	Smooth (like RBF)	Smooth (like RBF)	Smooth (like RBF)	Smooth (like RBF)	Smooth (like RBF)	Smooth (like RBF)
	High Lipschitz Continuity (like RBF)	High Lipschitz Continuity (like RBF)	High Lipschitz Continuity (like RBF)	High Lipschitz Continuity (like RBF)	High Lipschitz Continuity (like RBF)	High Lipschitz Continuity (like RBF)	High Lipschitz Continuity (like RBF)
	Similar to RBF Gradients	Similar to RBF Gradients	Similar to RBF Gradients	Similar to RBF Gradients	Similar to RBF Gradients	Similar to RBF Gradients	Similar to RBF Gradients
	Robust (Well-Conditioned Σ)	Robust (Well-Conditioned Σ)	Robust (Well-Conditioned Σ)	Robust (Well-Conditioned Σ)	Robust (Well-Conditioned Σ)	Robust (Well-Conditioned Σ)	Robust (Well-Conditioned Σ)
HMK	Piecewise Smooth	Piecewise Smooth	Piecewise Smooth	Piecewise Smooth	Piecewise Smooth	Piecewise Smooth	Piecewise Smooth
	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity	High Lipschitz Continuity
	Composite Gradients	Composite Gradients	Composite Gradients	Composite Gradients	Composite Gradients	Composite Gradients	Composite Gradients
	Highly Robust	Highly Robust	Highly Robust	Highly Robust	Highly Robust	Highly Robust	Highly Robust

and divergence measures.

Recommendations for Kernel Selection:

- **RBF Kernel**: Use when stability and simplicity are paramount, and the data exhibits smooth, isotropic patterns.
- **Polynomial Kernel**: Opt for when modeling high-degree interactions is essential, keeping in mind the need for careful parameter tuning.
- **Spectral Kernel**: Choose for applications involving graph-based data or scenarios requiring the capture of oscillatory relationships.
- **Mahalanobis Kernel**: Select when anisotropic similarity measures are necessary, ensuring that the covariance matrix is well-conditioned.
- **HMK**: Employ when leveraging the strengths of multiple kernels is advantageous, providing a balanced approach to handle both local and global dependencies in the data.

Designing kernels with favorable properties across different divergence measures is essential for robust and efficient optimization in alignment learning. Selecting the appropriate kernel based on the specific requirements of the task and the nature of the data can significantly enhance the performance and reliability of gradient-based learning algorithms.

I.29 Computational Overhead of DPO-Kernels

The computational complexity of DPO-Kernels stems from the integration of diverse kernels and divergence measures, each introducing unique bottlenecks and computational demands. This section provides an analysis of these costs based on kernel and divergence characteristics, as summarized in Tables 10 and 11.

Kernels: Balancing Flexibility and Cost The use of kernelized representations significantly enhances alignment flexibility but incurs varying degrees of computational and memory overhead:

- **RBF Kernel**: With a linear time and memory complexity of $O(m)$, the RBF kernel is efficient and widely applicable. It incurs a relative cost of 1.3× compared to standard DPO while maintaining high generalization. This makes it the default choice for tasks requiring fine-grained, local alignment.
- **Polynomial Kernel**: The computational cost of this kernel increases with its degree d, resulting in $O(m^d)$ complexity. While its relative cost ranges from 1.2–1.5×, its susceptibility to overfitting limits its generalizability, making it suitable for datasets with nonlinear dependencies.
- **Spectral Kernel**: Leveraging eigen decomposition or the Nyström method, this kernel achieves global structural alignment at the expense of $O(m^2)$ time and memory complexity. With a relative cost of 2–3×, it is ideal for tasks requiring

Kernel	Time Complexity	Memory Complexity	Key Bottleneck	Relative Cost (vs. DPO)	Generalization	Use Case
RBF	$O(m)$	$O(m)$	Euclidean distance computation	Low (1.3x)	High	Default Choice
Polynomial	$O(m^d)$	$O(m)$	Computation of $(u^\top v + c)^d$	Low (1.2-1.5x)	Risk of Overfitting	Nonlinear Datasets
Spectral	$O(m^2)$	$O(m^2)$	Eigen decomposition (or Nyström)	Medium (2-3x)	High	Global Structure
Mahalanobis	$O(m^3)$	$O(m^2)$	Inverting Σ and projection	High (3-5x)	High	Correlated Data
HMK	Depends on # of Kernels	Sum of each kernel's cost	Linear combination of kernels	Very High (3-4x)	Best	General Purpose

Table 10: Summary of Kernel Characteristics: Time Complexity, Memory Complexity, Bottleneck, Computational Cost, Generalization, and Use Cases.

Divergence	Time Complexity	Memory Complexity	Key Bottleneck	Relative Cost
KL Divergence	$\mathcal{O}(m)$	$\mathcal{O}(m)$	Logarithm and division on elements.	Low (1x)
Jensen-Shannon	$\mathcal{O}(m)$	$\mathcal{O}(m)$	Compute average distribution and KL.	Low (1.2x)
Wasserstein	$\mathcal{O}(m^3)$	$\mathcal{O}(m^2)$	Optimal transport (Sinkhorn) computation.	High (3-4x)
Rényi	$\mathcal{O}(m)$	$\mathcal{O}(m)$	Powers and divisions for each element.	Low (1.5x)
Bhattacharyya	$\mathcal{O}(m)$	$\mathcal{O}(m)$	Logarithmic computation of coefficient.	Low (1.3x)
Hellinger	$\mathcal{O}(m)$	$\mathcal{O}(m)$	Square root on probability elements.	Low (1.3x)
f-Divergence	$\mathcal{O}(m)$	$\mathcal{O}(m)$	Apply any convex function f to the ratio $\frac{p}{q}$.	Low (1.2x)

Table 11: Computational Cost Analysis for Divergence Functions.

broad, global relationships.

- **Mahalanobis Kernel**: The most computationally expensive kernel, with $O(m^3)$ time complexity, stems from the inversion of the covariance matrix. Its 3-5× relative cost is offset by robust generalization in tasks involving highly correlated data.
- **Hierarchical Mixture of Kernels (HMK)**: HMK dynamically combines local and global kernels, accumulating the individual costs of its components. While offering the best generalization, its computational demands are 3–4× higher than DPO for simple configurations, with costs escalating further depending on the number of kernels used.

Divergence Measures: Stability vs. Complexity The divergence regularizers integrated into DPO-Kernels introduce additional computational overhead, but they are generally less intensive than kernel operations:

- **Low-Cost Divergences**: KL divergence, Jensen-Shannon, Bhattacharyya, Rényi, and Hellinger divergences exhibit linear time complexity ($O(m)$), with relative costs ranging from 1× to 1.5×. These measures are efficient and versatile, suitable for most alignment tasks.
- **High-Cost Divergences**: Wasserstein divergence, requiring $O(m^3)$ time and $O(m^2)$ memory complexity, is the most computationally intensive due to optimal transport calculations. Despite its relative cost of 3–4×, it is highly effective for tasks involving significant distributional shifts.

Key Insights and Recommendations

- **Balancing Cost and Performance**: For resource-constrained settings, RBF and Polynomial kernels combined with low-cost divergences like KL or

Jensen-Shannon offer a pragmatic trade-off between computational efficiency and alignment performance.

- **Scalability of HMK**: The significant overhead of HMK necessitates exploration of approximation techniques like Random Fourier Features (RFF) or Nyström methods to reduce computational demands while preserving performance.
- **Task-Specific Optimization**: Divergence selection should align with task complexity, leveraging efficient measures like Hellinger for standard alignment and Wasserstein for complex distributional shifts.

Addressing these computational challenges is critical for scaling DPO-Kernels to real-world, multimodal, and large-scale alignment tasks (Tables 10 and 11).

J Results & Analysis

This section provides a detailed analysis of the performance of our proposed approach, focusing on the efficacy of Hybrid Loss, the role of Divergence-based regularizers, and the impact of Safety Fine-Tuning. Each subsection delves into the quantitative and qualitative aspects of the proposed methods, supported by theoretical analysis and empirical results.

J.1 Efficacy of Hybrid Loss

Motivation and Design: The Hybrid Loss is designed to combine the benefits of probability-based loss (which focuses on probability alignment) and embedding-based loss (which captures structural alignment). By leveraging both perspectives, the Hybrid Loss aims to achieve better generalization, especially in tasks where alignment requires multi-scale adaptation.

Theoretical Justification: The Hybrid Loss $\mathcal{L}_{\text{Hybrid}}$ is defined as:

$$\mathcal{L}_{\text{Hybrid}} = \alpha\,\mathcal{L}_{\text{Probability}} + (1-\alpha)\,\mathcal{L}_{\text{Embedding}}$$

where $\alpha \in [0, 1]$ is a learnable coefficient that controls the balance between the two components. The probability-based loss is effective for fine-grained preference alignment, while the embedding loss captures semantic structure.

Empirical Evidence: We conduct experiments across 13 datasets with varying levels of complexity (e.g., factuality, reasoning, and safety). Fig. 18 shows the performance of Hybrid Loss compared to baseline methods. The Hybrid Loss consistently outperforms both standalone probability and embedding losses, with an average relative improvement of 9.2%. This demonstrates the complementary nature of the two loss components. Fig. 18 shows the F1 scores of the RBF kernel with divergence-based regularizers across key tasks.

J.2 Efficacy of Divergence-Based Regularizers

Overview: Divergence-based regularizers are crucial in aligning model-generated distributions with human-preferred distributions. We explore several divergence measures, including Kullback-Leibler (KL), Jensen-Shannon (JS), Hellinger, Rényi, Bhattacharyya, and Wasserstein divergences.

Mathematical Formulation: Given two distributions P and Q, the divergence-based regularization term $\mathcal{R}_{\text{Divergence}}$ is defined as:

$$\mathcal{R}_{\text{Divergence}} = \sum_{i=1}^{n} D(P_i \| Q_i)$$

where $D(P\|Q)$ can be any of the aforementioned divergence measures.

Empirical Analysis: To evaluate the efficacy of each divergence, we measure the alignment score (AS) on multiple datasets. Fig. 20 illustrates that Wasserstein divergence achieves the best performance due to its ability to consider distance in the probability space, unlike KL or JS which may suffer from zero-probability issues.

Takeaway: Wasserstein divergence offers the most consistent performance gains across datasets. This supports the claim that Wasserstein's ability to

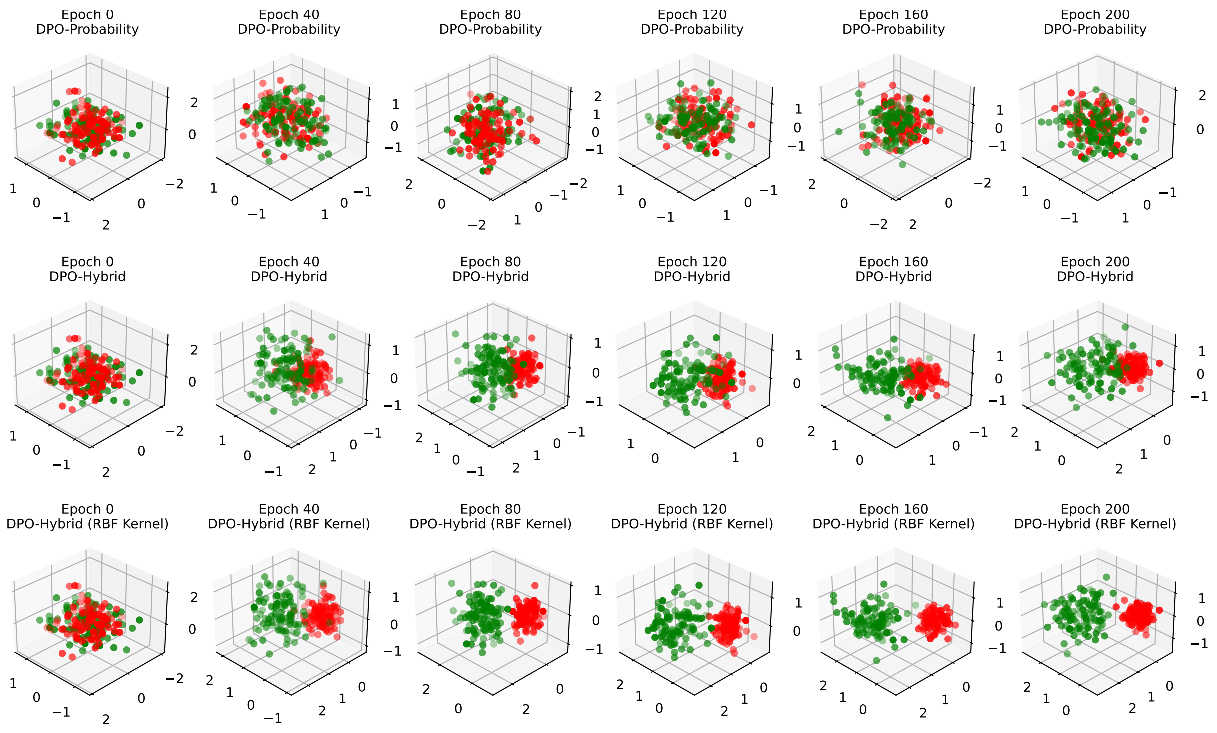


Figure 17: **Evolution of LLM Logits Across Epochs for DPO-Probability Loss, DPO-Hybrid Loss, and DPO-Hybrid (RBF Kernel) Loss.** This figure presents the evolution of logits, treated as embeddings, at six key epochs (0, 40, 80, 120, 160, 200) for three alignment methods: DPO-Probability Loss, DPO-Hybrid Loss, and DPO-Hybrid (RBF Kernel) Loss. The logits are projected into a 3D space using t-SNE (van der Maaten and Hinton, 2008) applied to the alignment space, where red points represent *rejected samples* and green points represent *selected samples*. At epoch 0, all methods share identical embeddings. As training progresses, DPO-Probability Loss shows modest clustering improvements. In contrast, DPO-Hybrid Loss achieves better separation between selected and rejected samples, with notable improvements after epoch 80. The DPO-Hybrid (RBF Kernel) Loss achieves the most pronounced clustering, with significantly tighter and more distinct groupings of red and green points due to the enhanced capacity of RBF kernels to model nonlinear separations. This visualization highlights the superior alignment capabilities of DPO-Hybrid (RBF Kernel) Loss compared to DPO-Hybrid Loss and DPO-Probability Loss.

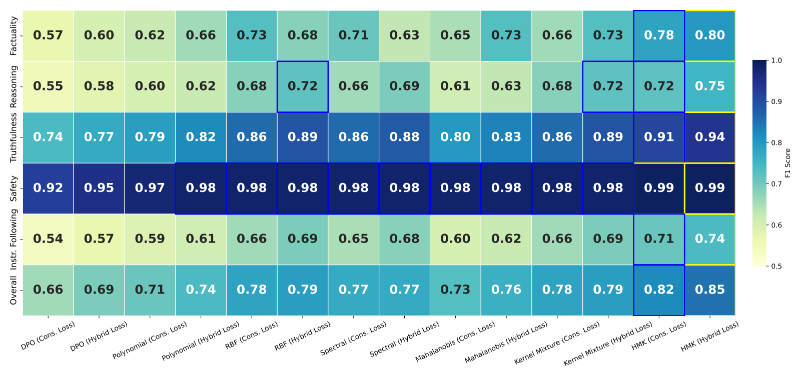


Figure 18: Heatmap depicting F1 scores across various kernels and loss functions for alignment tasks. The yellow borders indicate the best-performing kernels for each task, while blue borders highlight the second-best performers. Scores are evaluated for tasks such as Factuality, Reasoning, Truthfulness, Safety, and Instruction Following, with an overall assessment summarized in the last row. The HMK (Hybrid Loss) kernel consistently demonstrates top performance in multiple tasks.

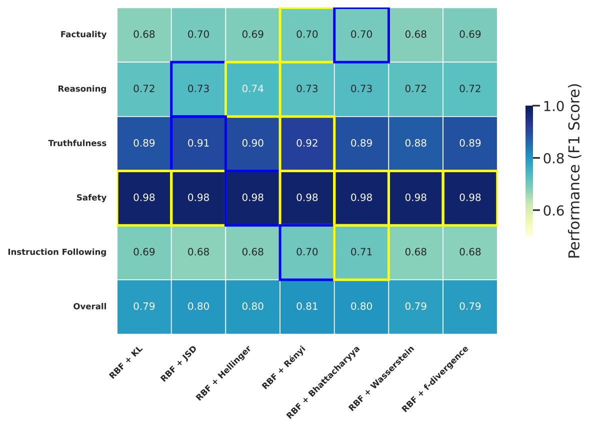


Figure 19: F1 scores of the RBF kernel with divergence-based regularizers across key tasks. Results for all kernel-divergence combinations are detailed in Appendix J.

model the "distance" between distributions makes it more suitable for alignment tasks than the KL or JS divergences.

K Gradient Descent Dynamics on Kernel-Induced Loss Landscapes

In this section, we analyze the gradient descent dynamics on loss landscapes induced by four widely-used kernels: **RBF**, **Polynomial**, **Spectral**, and **Mahalanobis**. Using gradient vector fields and contour plots, we highlight the key optimization behaviors associated with each kernel. These insights elucidate why certain kernels are more effective for stable and efficient optimization in machine learning tasks.

K.1 Visualization of Gradient Fields and Contour Plots

Figure 21 illustrates the gradient fields overlaid on the contour plots for the loss landscapes induced by the four kernels. The following observations provide insights into the behavior of each kernel:

- **RBF Kernel**:
  - **Smoothness**: The contours are isotropic, forming circular basins of attraction.
  - **Gradient Behavior**: Gradients guide the parameters steadily toward the global minimum, promoting stable and fast convergence.
  - **Suitability**: Ideal for tasks with smooth and convex loss landscapes, as supported by theoretical guarantees for convergence (Schölkopf and Smola, 2002).
- **Polynomial Kernel**:

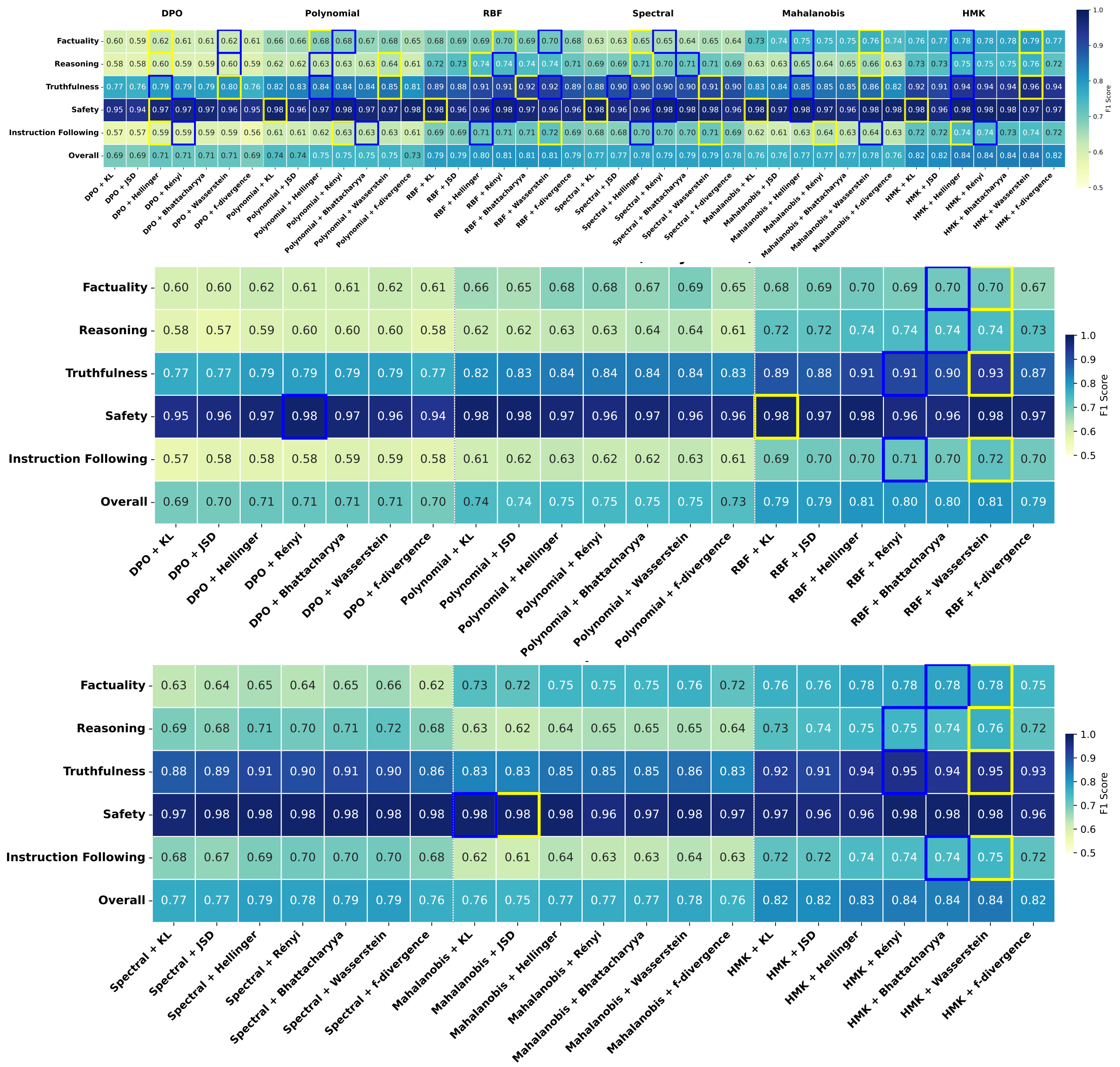


Figure 20: Heatmaps illustrating the performance of kernel-divergence combinations across alignment tasks. The first heatmap presents the complete view, showcasing all kernels (DPO, Polynomial, RBF, Spectral, Mahalanobis, HMK) paired with divergences (KL, JSD, Hellinger, Rényi, Bhattacharyya, Wasserstein, f-divergence). The second and third heatmaps split the data for clarity, focusing on the first three kernels (DPO, Polynomial, RBF) and the last three kernels (Spectral, Mahalanobis, HMK), respectively. Each row represents a task (Factuality, Reasoning, Truthfulness, Safety, Instruction Following), while the "Overall" row aggregates average performance. Yellow and blue borders highlight the best and second-best-performing kernel-divergence combinations for each task.

- **Non-Convexity**: The contours exhibit sharp transitions and irregular regions, creating multiple local minima.
- **Gradient Behavior**: Gradients are chaotic in regions with high curvature, causing sensitivity to initialization.
- **Suitability**: Effective for problems requiring higher-order feature interactions, though sensitive to hyperparameter choices like degree d (Shawe-Taylor and Cristianini, 2004).

- **Spectral Kernel**:

- **Oscillatory Nature**: The contours are highly dependent on the choice of basis functions ϕ_i, resulting in oscillations.
- **Gradient Behavior**: Gradients exhibit abrupt changes in direction, slowing convergence in regions of high oscillation.
- **Suitability**: Useful for data with inherent periodicity or hierarchical structures (Ng et al., 2001).

- **Mahalanobis Kernel**:

- **Anisotropy**: The contours are elongated along certain directions, determined by the covariance matrix Σ.
- **Gradient Behavior**: Gradients are well-aligned with the anisotropic structure, ensuring robust convergence when Σ is well-conditioned.
- **Suitability**: Well-suited for tasks with correlated features or structured data distributions (Weinberger and Saul, 2009).

K.2 Insights from Gradient Fields

- **Smoothness and Stability**: RBF and Mahalanobis kernels provide smooth and stable landscapes, favoring robust and fast convergence. Polynomial and Spectral kernels introduce non-convexity and oscillations, requiring careful initialization and hyperparameter tuning.

- **Directional Dependencies**: The Mahalanobis kernel adapts to feature correlations through Σ, whereas the RBF kernel provides isotropic behavior. Spectral kernel gradients align with the chosen basis functions, offering flexibility at the cost of stability.

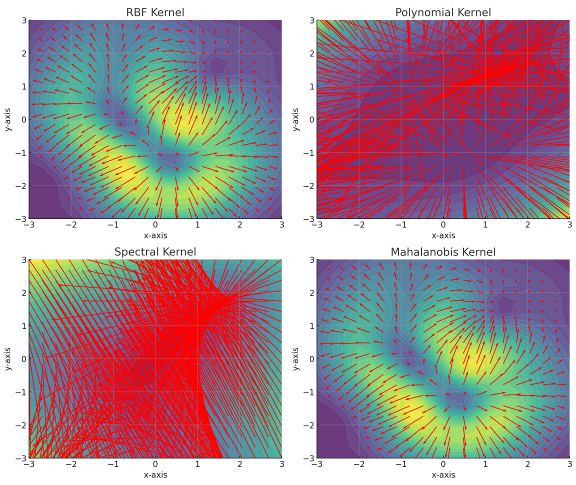


Figure 21: Contour plots overlaid with gradient descent fields for different kernels. Each plot illustrates the gradient dynamics and loss landscape for the respective kernels: (Top-left) RBF Kernel, showing smooth, isotropic gradients guiding efficient convergence; (Top-right) Polynomial Kernel, exhibiting sharp transitions and chaotic gradients in non-convex regions; (Bottom-left) Spectral Kernel, characterized by oscillatory contours and abrupt gradient changes aligned with basis functions; (Bottom-right) Mahalanobis Kernel, demonstrating anisotropic gradients aligned with the covariance structure, ensuring robust optimization when Σ is well-conditioned. Red arrows represent the gradient vectors, highlighting the direction and intensity of optimization steps across the loss landscape.

- **Optimization Challenges**: Polynomial and Spectral kernels require additional regularization or initialization strategies to mitigate sensitivity to local minima and oscillations.

The contour plots and gradient fields reveal how kernel-induced loss landscapes shape gradient descent dynamics. The RBF and Mahalanobis kernels are robust choices for stable optimization, while Polynomial and Spectral kernels provide flexibility at the expense of increased sensitivity and potential instability. These insights underscore the importance of kernel selection in achieving efficient and effective optimization for diverse machine learning tasks.

L Mechanism of Safety Fine-Tuning: Safe vs. Unsafe Cluster Effects

Jain et al. (2024b) demonstrate that safety fine-tuning (alignment) minimally adjusts MLP weights in LLMs to project unsafe inputs into the null space of weight matrices, inducing distinct clustering of inputs based on safety status. We analyze the evolution of these clusters during training and evaluate their separation using the Davies-Bouldin Score (DBS), where lower values indicate better clustering with compact intra-cluster distances and large inter-cluster separations.

Definition: For k clusters $\{C_1, C_2, \ldots, C_k\}$, DBS (Davies and Bouldin, 1979) is defined as:

$$DBS = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{S_i + S_j}{D_{ij}} \right),$$

where:

- $S_i = \frac{1}{|C_i|} \sum_{x \in C_i} \|x - \mu_i\|$: Average intra-cluster distance for cluster C_i, with μ_i as its centroid.
- $D_{ij} = \|\mu_i - \mu_j\|$: Distance between centroids of clusters C_i and C_j.

Lower DBS values in alignment learning indicate:

- **Clearer Decision Boundaries:** Better separation of safe and unsafe clusters for precise behavior control.
- **Improved Generalization:** Enhanced performance on unseen data through well-separated clusters.
- **Increased Robustness:** Compact clusters with strong separation reduce sensitivity to noise and outliers. cf. sec:appendix:safe_unsafe_cluster.

?? visualizes the kernel embeddings after 200 epochs across different kernels: Polynomial, Spectral, RBF, Mahalanobis, and HMK. Green points represent selected samples, while red points indicate rejected samples, illustrating how each kernel processes the data. The RBF and HMK kernels demonstrate strong separation between selected and rejected samples, highlighting their superior alignment performance. In contrast, the Polynomial and Mahalanobis kernels exhibit less distinct separation.

M Score-Based Analysis of Cluster Separation

Jain et al. (2024b) show that safety fine-tuning aka alignment minimally adjusts MLP weights in LLMs to project unsafe inputs into the null space of the weight matrices. This process induces a distinct clustering of inputs, separating them based on safety status. Our analysis focuses on how these clusters evolve during training and evaluates their separation using the **Davies-Bouldin Score (DBS)**, a standard metric for cluster quality. Lower DBS values indicate better clustering, characterized by compact intra-cluster distances and large inter-cluster separations.

M.1 Introduction to Davies-Bouldin Score (DBS)

The **Davies-Bouldin Score (DBS)** is a widely adopted metric in unsupervised and semi-supervised learning for evaluating clustering performance (Davies and Bouldin, 1979). It effectively measures the balance between intra-cluster compactness and inter-cluster separation. A lower DBS is preferable, as it implies that clusters are both tightly packed and well-separated.

M.1.1 Definition

For a set of k clusters $\{C_1, C_2, \ldots, C_k\}$, the DBS is mathematically defined as:

$$DBS = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{S_i + S_j}{D_{ij}} \right)$$

where:

- S_i: Average intra-cluster distance for cluster C_i, given by:

$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} \|x - \mu_i\|$$

  where μ_i is the centroid of cluster C_i.

- D_{ij}: Distance between the centroids of clusters C_i and C_j, calculated as:

$$D_{ij} = \|\mu_i - \mu_j\|$$

M.1.2 Intuition Behind DBS

The DBS provides key insights into the clustering process:

- **Diffuse Clusters:** A high intra-cluster scatter (S_i) results in a high DBS, penalizing poorly formed clusters.
- **Cluster Overlap:** A low inter-cluster distance (D_{ij}) increases the DBS, penalizing clusters that are too close to one another.

In the context of alignment learning, a lower DBS is critical for achieving:

- **Clearer Decision Boundaries:** Better separation between safe and unsafe clusters enables more precise behavior control.
- **Improved Generalization:** Well-separated clusters reduce ambiguities, enhancing model performance on unseen data.
- **Increased Robustness:** Compact and well-separated clusters are less sensitive to outliers and noisy data.

M.1.3 Results Analysis

The evaluation of the three alignment approaches—DPO-Probability Loss, DPO-Hybrid Loss, and DPO-Hybrid (RBF Kernel) Loss—shows distinct cluster behaviors across training epochs. The results are quantified using DBS and reported in Table 12, Figure 17 visually summarizes clustering effect accross DPO-Probability Loss, DPO-Hybrid Loss, and DPO-Hybrid (RBF Kernel) Loss. As training progresses, the DPO-Hybrid (RBF Kernel) achieves the lowest DBS, reflecting its superior ability to distinguish between safe and unsafe clusters.

Table 12: Cluster Separation Measured by Davies-Bouldin Score for DPO Methods (Lower is Better).

Epochs	DPO-Probability	DPO-Hybrid	DPO-Hybrid (RBF Kernel)
0	2.15	2.08	2.01
40	1.94	1.84	1.75
80	1.75	1.62	1.43
120	1.62	1.43	1.20
160	1.45	1.26	1.10
200	1.32	1.15	0.92

To further assess the generalization capabilities, we analyzed five kernel types—Polynomial, Spectral, RBF, Mahalanobis, and Hierarchical Mixture of Kernels (HMK)—across epochs. Table 13 captures the DBS results for each kernel, highlighting the exceptional performance of HMK, which consistently achieves the lowest scores, signifying compact and well-separated clusters. THe Findings is visually summarized in Figure 22.

Table 13: Cluster Separation Measured by Davies-Bouldin Score for Kernel Methods (Lower is Better).

Epochs	Polynomial	Spectral	RBF	Mahalanobis	HMK
0	2.25	2.10	2.01	2.02	1.90
40	2.12	1.95	1.84	1.85	1.65
80	1.95	1.80	1.65	1.63	1.28
120	1.85	1.65	1.45	1.40	1.05
160	1.72	1.50	1.25	1.20	0.95
200	1.60	1.35	1.10	1.05	0.80

N Heavy-Tailed Self-Regularization (HT-SR) Theory and Generalization

The Heavy-Tailed Self-Regularization (HT-SR) theory provides a statistical mechanics framework to analyze the weight matrices of Deep Neural Networks (DNNs). It demonstrates that the eigenvalue spectra of the weight matrices often follow

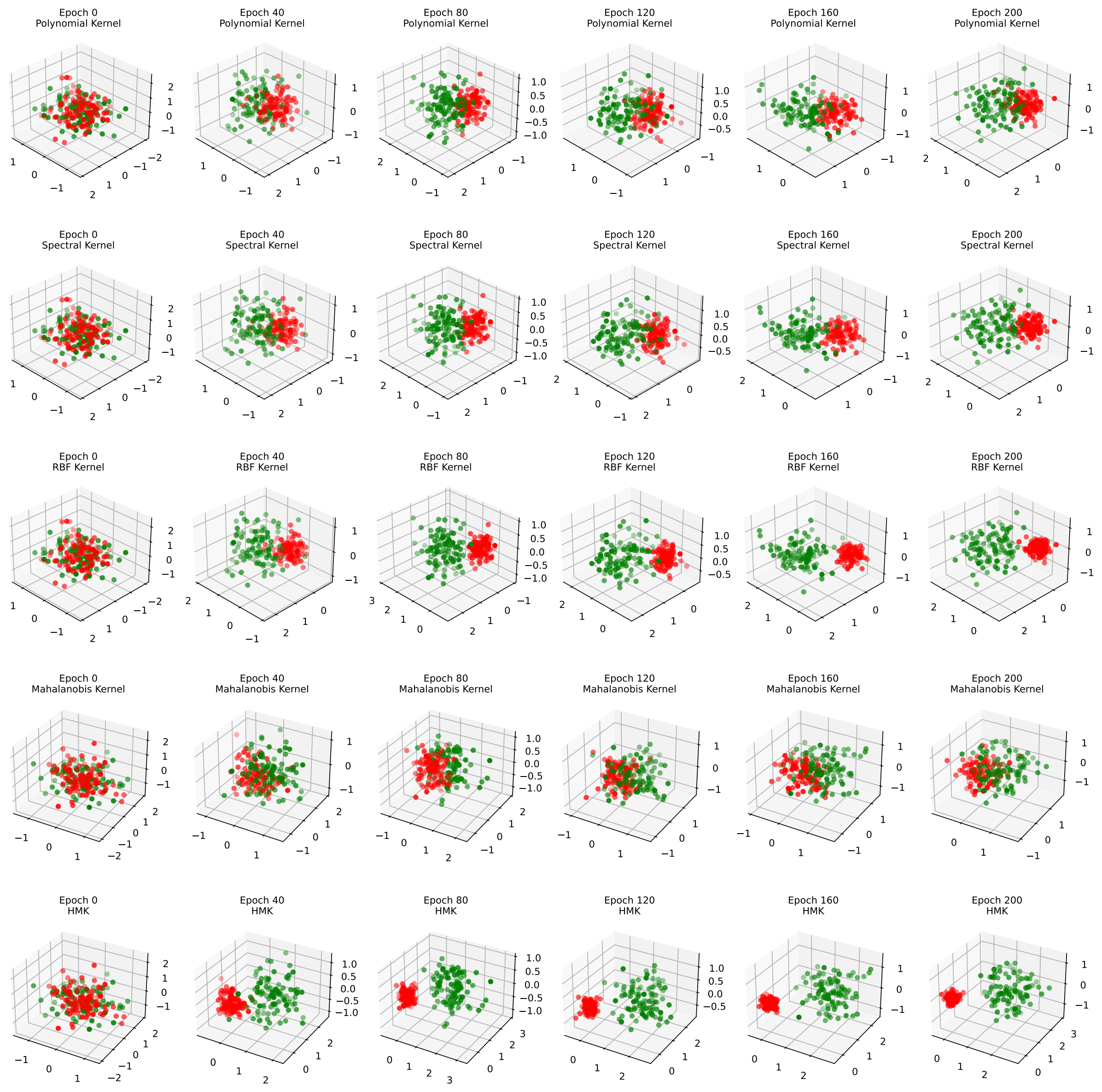


Figure 22: Visualization of the embedding evolution of five kernel types—Polynomial, Spectral, RBF, Mahalanobis, and Hierarchical Mixture of Kernels (HMK)—over six training epochs (0, 40, 80, 120, 160, and 200). Each row represents the clustering progression for a specific kernel, with the red points indicating rejected samples and the green points representing selected samples. The HMK demonstrates superior clustering capabilities compared to the other kernels, exhibiting more compact and well-separated clusters. This visualization highlights the relative clustering effectiveness of each kernel across epochs, with HMK achieving the most distinct and organized separation.

heavy-tailed distributions, which are indicative of self-organized criticality and implicit regularization during optimization. This behavior suggests that the weight matrices capture correlations across multiple scales, which is a key factor in enhancing generalization capabilities (Martin et al., 2021b).

N.1 Core Insights of HT-SR Theory

1. **Empirical Spectral Density (ESD):** The eigenvalue distribution $\rho(\lambda)$ of a weight matrix $\mathbf{W}$ is given by:

$$\rho(\lambda) = \frac{1}{N} \sum_{i=1}^{N} \delta(\lambda - \lambda_i), \tag{3}$$

where $\{\lambda_i\}$ are the eigenvalues of $\mathbf{W}^\top \mathbf{W}$. HT-SR theory posits that $\rho(\lambda)$ often follows a truncated power law:

$$\rho(\lambda) \propto \lambda^{-\alpha}, \quad \text{for } \lambda_{\text{min}} \leq \lambda \leq \lambda_{\text{max}}. \tag{4}$$

The exponent α characterizes the tail behavior, with smaller α values ($\alpha \in [2, 4]$) correlating with better generalization.

2. **Weighted Alpha Metrics:** HT-SR introduces the *Weighted Alpha*, computed as:

$$\alpha_w = \frac{\sum_{i=1}^{N} \lambda_i^{-\alpha} \log(\lambda_i)}{\sum_{i=1}^{N} \lambda_i^{-\alpha}}, \tag{5}$$

and the Log α-Norm:

$$\text{Log-}\alpha = \frac{1}{N} \sum_{i=1}^{N} \log(\lambda_i). \tag{6}$$

These metrics serve as robust predictors of model quality, outperforming traditional norm-based measures, especially in differentiating well-trained versus poorly trained models.

3. **Correlation Flow:** Stable α values across network layers suggest "Correlation Flow," where features propagate effectively through the network. For weight matrices $\mathbf{W}_l$ at layer l, HT-SR ensures α_l remains within the optimal range, preserving consistent feature extraction:

$$\alpha_l \approx \text{constant}, \quad \forall l \in \{1, \ldots, L\}. \tag{7}$$

N.2 Implications for Generalization

HT-SR theory highlights that well-trained models exhibit eigenvalue spectra with heavy-tailed distributions. Models with excessively large α values may be over-parameterized or poorly trained, as their weight matrices lack the desired multi-scale correlation structure. In contrast, weight matrices with optimal α values achieve better generalization by implicitly balancing expressiveness and complexity.

N.3 Empirical Validation in DNNs

Empirical studies on architectures like ResNet, DenseNet, and GPT validate HT-SR theory: - **ResNet:** Deeper models exhibit smaller and more stable α values, which correlate strongly with improved test accuracy and generalization. - **DenseNet:** The excessive connectivity in DenseNet models leads to less favorable spectral properties, with higher α values indicating suboptimal performance.

For instance, models with $\alpha \approx 2.5$ consistently outperform those with $\alpha \geq 5$ on tasks requiring robust generalization.

N.4 Applications in Pretrained Models

HT-SR metrics enable model quality assessments without training or test data by analyzing eigenvalue spectra. This is particularly valuable for pretrained models, allowing: - Detection of "Scale Collapse," where spectral norms deviate anomalously. - Fine-tuning guidance based on layer-wise spectral analysis.

N.5 Conclusion and Future Directions

HT-SR theory bridges the gap between statistical mechanics and machine learning by linking implicit regularization to generalization. Future research could explore: - Extending HT-SR to unsupervised and reinforcement learning settings. - Refining HT-SR metrics for real-time model diagnostics and training stabilization.

O Hyperparameters and Best Practices

This section summarizes the hyperparameters used in our approach and provides best practices for their configuration. Table 14 outlines the recommended ranges, descriptions, and practical guidelines for each hyperparameter. These recommendations are derived from empirical experiments and theoretical insights, aiming to optimize performance across diverse alignment tasks.

The hyperparameters are categorized based on their roles, such as kernel configuration, regularization, and alignment strategies. For instance, α and β control the trade-off between alignment robustness and regularization strength, while τ determines the balance between local and global kernel contributions in HMK. Proper tuning of these hyperparameters is crucial for achieving compact and well-separated clusters, as evidenced by the Davies-Bouldin score analysis in previous sections.

O.1 Approaches for Hyperparameter Selection

Effective hyperparameter selection is crucial for ensuring the optimal performance of DPO-Kernels and Hierarchical Mixture of Kernels (HMK). Key hyperparameters include the RBF bandwidth σ, Polynomial degree d, Mahalanobis covariance Σ, and mixture weights λ_i. Below, we outline practical approaches for hyperparameter selection and tuning.

O.1.1 1. Random Search and Grid Search

Random search and grid search are standard approaches for hyperparameter tuning (Bergstra and Bengio, 2012). While grid search explores a fixed set of values, random search samples from a distribution, often achieving better results with fewer trials.

Best Practices:

- **RBF Bandwidth σ**: Sample σ from a logarithmic scale, e.g., $\sigma \in [10^{-3}, 10^{3}]$, as sensitivity to changes in σ is non-linear.
- **Polynomial Degree d**: Use small integer degrees $d \in \{2, 3, 4, 5\}$ to avoid excessive non-convexity.
- **Mixture Weights λ_i**: Use Dirichlet-distributed samples to ensure $\sum_i \lambda_i = 1$.

O.1.2 2. Bayesian Optimization

Bayesian optimization (BO) models the loss as a Gaussian process and efficiently balances exploration and exploitation (Snoek et al., 2012). BO identifies the optimal hyperparameters by maximizing the Expected Improvement (EI).

Mathematical Formulation:

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} \mathrm{EI}(\boldsymbol{\lambda}),$$

where $\mathrm{EI}(\boldsymbol{\lambda})$ is the expected improvement over the best observed loss. Bayesian optimization is useful for tuning computationally expensive hyperparameters like Mahalanobis covariance Σ.

Best Practices:

- Use multi-fidelity optimization to reduce computational costs (Li et al., 2018).
- Apply BO for **non-differentiable hyperparameters** (e.g., Polynomial degree d and kernel mixture weights λ).

O.1.3 3. Cross-Validation

Cross-validation is a robust strategy to tune hyperparameters, especially for ensuring generalization (Koh et al., 2021b). For each hyperparameter configuration, k-fold cross-validation partitions the data into k folds, trains on $k-1$ folds, and evaluates on the remaining fold.

Mathematical Formulation:

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(\boldsymbol{\lambda}, \mathcal{D}_i),$$

where $\mathcal{L}(\boldsymbol{\lambda}, \mathcal{D}_i)$ is the loss on the i-th fold. Cross-validation is particularly effective for selecting global hyperparameters like kernel types and mixture coefficients λ_i.

Table 14: Summary of Hyperparameters and Best Practices

Hyperparameter	Description	Recommended Range	Best Practices
α (Alpha)	Controls the strength of the regularization (alignment with reference policy).	$0.1 \leq \alpha \leq 1.0$	Start with $\alpha = 0.5$ for balanced flexibility and conservativeness. Lower values allow greater personalization.
β (Beta)	Scaling factor for divergence-based regularizers.	$0.5 \leq \beta \leq 2.0$	Increase β for stronger penalization of distributional deviations; tune based on task complexity.
γ (Gamma)	Weight for embedding-based alignment signals.	$0.1 \leq \gamma \leq 1.0$	Use $\gamma > 0.5$ for semantic alignment; lower values emphasize probability-based preferences.
Kernel Mixture Weights	Weights for Polynomial, RBF, Spectral, and Mahalanobis kernels.	Sum to 1.0, individually > 0.1	Initialize evenly (0.25 each) or based on data insights. Dynamically learned during training.
σ (Sigma)	Bandwidth parameter for RBF kernel.	$0.1 \leq \sigma \leq 2.0$	Lower σ sharpens RBF locality. Tune with cross-validation based on data density.
d (Degree)	Degree of Polynomial kernel.	$2 \leq d \leq 5$	Start with $d = 2$ for efficiency. Higher values capture complex interactions but may risk overfitting.
λ (Lambda) Divergences	Weights for divergence terms (e.g., JS, Wasserstein, Bhattacharyya).	Sum to 1.0, individually > 0.1	Prioritize Wasserstein or Bhattacharyya for safety tasks and JS for semantic alignment.
τ (Tau)	Balance between local and global kernel contributions in HMK.	$0.3 \leq \tau \leq 0.7$	Use $\tau = 0.5$ for balanced contributions. Adjust based on alignment needs (e.g., $\tau > 0.5$ for finer local adjustments).
Effective Range (r)	Defines the influence zone of kernels like RBF and Mahalanobis.	Task-dependent	Align σ or Σ regularization to optimize locality versus global correlation capture.
Embedding Similarity Scaling	Scaling factor for embedding-based pairwise metrics.	$0.5 \leq$ scale ≤ 1.5	Normalize embedding spaces before applying similarity metrics. Cross-validate scaling on validation data.
Regularizer Thresholds	Thresholds for divergence-specific terms (e.g., Rényi's α, support overlap).	$0.1 \leq$ threshold ≤ 0.6	Tighter thresholds improve separation but may increase computational cost.

O.1.4 4. Adaptive Hyperparameter Selection

For hyperparameters like mixture weights τ_1, τ_2 in HMK, it is beneficial to adaptively learn them during training via backpropagation. Differentiable hyperparameters can be updated using gradient-based methods.

Mathematical Formulation:

$$\lambda_{t+1} = \lambda_t - \eta \nabla_\lambda \mathcal{L}(\boldsymbol{\lambda}; \mathcal{D}),$$

where η is the learning rate and $\nabla_\lambda \mathcal{L}$ is the gradient of the loss with respect to λ. This approach enables dynamic adaptation of kernel mixture weights and bandwidths during training.

O.1.5 5. Early Stopping

Early stopping halts training once the validation loss no longer improves. This is particularly useful for adjusting learning rates, mixture weights, and other training-related hyperparameters (Prechelt, 1998).

Best Practices:

- Monitor the validation loss for p epochs and stop training if no improvement is observed.
- Early stopping can also be used to tune the kernel mixture weights τ_1, τ_2 during training.

We have presented five key approaches for hyperparameter selection in DPO-Kernels and HMK, including random/grid search, Bayesian optimization, cross-validation, adaptive tuning, and early stopping. Bayesian optimization and cross-validation are ideal for non-differentiable hyperparameters, while adaptive methods are effective for differentiable hyperparameters like mixture weights τ_1 and τ_2. Future research could incorporate meta-learning (Finn et al., 2017b) to automate hyperparameter selection for DPO-Kernels.