Distribution-Aligned Decoding for Efficient LLM Task Adaptation

Senkang Hu[1,2,*], **Xudong Han**[3,*], **Jinqi Jiang**[4], **Yihang Tao**[1,2], **Zihan Fang**[1,2]
Yong Dai[5], **Sam Tak Wu Kwong**[6], **Yuguang Fang**[1,2,†]
[1]Hong Kong JC STEM Lab of Smart City, [2]City University of Hong Kong, [3]University of Sussex,
[4]Huazhong University of Science and Technology, [5]Fudan University, [6]Lingnan University
{senkang.forest, yihang.tommy, zihanfang3-c}@my.cityu.edu.hk,
xh218@sussex.ac.uk, jinqijiang667@gmail.com, daiyongya@outlook.com,
samkwong@ln.edu.hk, my.Fang@cityu.edu.hk

Background

Why do We Still Chase the Weights?

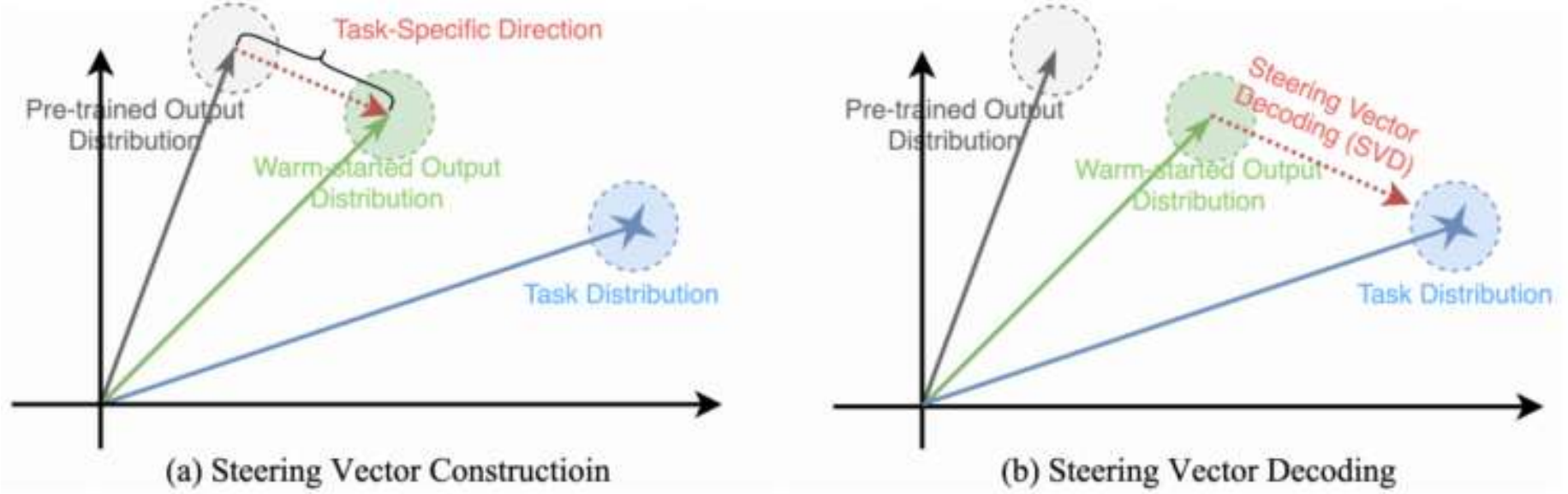
The end goal of adaptation is to shift the model's output distribution so that $P_\theta(y|x)$ aligns with the task-specific target.

However, there are three problems:

a) training scales linearly in model size and data epochs;

b) weight updates can have unpredictable, non-local effects on token probabilities;

c) a fixed hyper-parameter often fails to transfer across tasks and domains.



Steering Vector Construction

Warm-Start

In order to construct the steering vector, we first need to know the task-specific direction of the steering vector.

Given a pre-trained LLM with the parameter θ, the model defines conditional probability $P_\theta(y|x)$ over output text y given input x. If $y = (y^1, \ldots, y^T)$ is a sequence of T tokens, this typically factorizes autoregressively as:$P_\theta(y|x) = \prod_{t=1}^{T} P_\theta(y^t|x, y^{<t})$, where $y^{<t}$ denotes the sequence of previous tokens, x is the input tokens.

Given a downstream dataset $D_{task} = \{(x_i, y_i)\}_{i=1}^{N}$, then we warm-start the model by fine-tuning one epoch in D_{task} or part of the dataset. The model can be formulated as $P_\Phi(y|x)$, where Φ is the updated parameters, and we believe that the model's output distribution $P_\Phi(y|x)$ is close to the task-specific target distribution $P_{task}(y|x)$ compared with the pre-trained distribution $P_\theta(y|x)$ since the warm-started model's training loss decreases and the test accuracy increases.

Steering Vector Construction

KL Gradient as Steering Signal

If we use $KL(P_\Phi(y|x)||P_\theta(y|x))$ to measure the difference, it means that we assume that the warm-started model knows more about the task than the pre-trained model, and we want to steer the model's output distribution towards the task-specific target distribution.

$$KL(P_\Phi(y|x)||P_\theta(y|x)) = \sum_{y\in Y} P_\Phi(y|x) log \frac{P_\Phi(y|x)}{P_\theta(y|x)}$$

We compute the gradient with respect to by taking the partial derivative of KL with respect to each P_Φ

$$\frac{\partial KL}{\partial P_{\Phi,y_i}} = \frac{\partial}{\partial P_{\Phi,y_i}}(P_{\Phi,y_i} log P_{\Phi,y_i} - P_{\Phi,y_i} log P_{\theta,y_i}) = log(\frac{P_{\Phi,y_i}}{P_{\theta,y_i}}) + 1$$

Therefore, the gradient $\nabla_{P_\Phi} KL(P_\Phi||P_\theta)$ is a vector

$$\nabla_{P_\Phi} KL(P_\Phi||P_\theta) = [log(\frac{P_{\Phi,y_1}}{P_{\theta,y_1}}) + 1, log(\frac{P_{\Phi,y_2}}{P_{\theta,y_2}}) + 1, \ldots, log(\frac{P_{\Phi,y_{|Y|}}}{P_{\theta,y_{|Y|}}}) + 1]$$

if we use the negative of this gradient as our steering vector, it represents the direction of task-specific knowledge that the warm-started model has acquired relative to the pre-trained model.

Steering Vector Construction

Logic-Space Projection

The simplest approach is to directly apply this vector to adjust the decoding distribution

$$\hat{P}(y|x) = (1-\mu) * P_{\Phi}(y|x) + \mu * (-\nabla_{P_{\Phi}} KL(P_{\Phi}||P_{\theta}))$$

However, this method introduces several practical issues:

a) $P_{\Phi}(y|x)$is a probability distribution over the vocabulary, the adjusted distribution $\hat{P}$ must satisfy $\sum_y \hat{P}(y|x) = 1$;

b) The gradient involves logarithmic terms $log\, {}^{P_{\Phi}(y)}/_{P_{\theta}(y)}$, which can be numerically unstable when $P_{\Phi}(y)$ or $P_{\theta}(y)$ are close to zero;

c) The KL gradient is defined in the Euclidean tangent space of the probability simplex. Without proper geometric projection, applying this vector may lead to invalid probability values.

To resolve this, we shift to the logic space.

Steering Vector Construction

To resolve this, we can project the KL gradient from probability space into logit space via the softmax Jacobian matrix by leveraging the chain rule:

$$\delta_{logits} = J * (-\nabla_{P_\Phi} KL(P_\Phi || P_\theta)) = (diag(P_\Phi) - P_\Phi {P_\Phi}^T) * (-log \frac{P_\Phi}{P_\theta} - 1)$$

This projected vector δ_{logits} serves as a task-aware logit delta, which can be added to the original logits before softmax:

$$\hat{z}_\Phi = z_\Phi + \mu * \delta_{logits}, \qquad \hat{P} = Softmax(\hat{z}_\Phi)$$

Althought δ_{logits} captures the KL gradient direction in logit space, it can still be dominated by false positive tokens-tokens that are not semantically relevant but receive large KL gradients due to numerical instability, e.g., when P_θ is extremely small.

Steering Vector Construction

Confidence-Aware Steering Vector Constraint

Define the confidence of each token $y \in \mathrm{V}$ at a decoding step as its predicted probability under the task-adapted model $s(y) = P_{\Phi}(y|x)$. Let $y^* = argmax_{y \in \mathrm{V}} P_{\Phi}(y|x)$ denote the most likely token. Then we introduce a threshold $\alpha \in (0,1]$ to retain only the confident tokens which have a probability greater than α times the probability of the most likely token. The binary mask $I(y)$ is defined as:

$$I(y) = 1(P_{\Phi}(y) \geq \alpha * P_{\Phi}(y^*))$$

We now mask the logit delta by element-wise applying the confidence mask and penalty:

$$\hat{\delta}_{logits}(y) = I(y) * \delta_{logits} + (1 - I(y)) * \lambda$$

where λ is a constant penalty term.

Task-Aware Steering Vector Decoding

Logit Adjustment

First, compute the logit $z_\Phi(y)$ for each token $y \in \mathrm{V}$ using the task-adapted model $P_\Phi(y|x)$.

Then, apply the task-aware steering vector with the condifence mask constraint to steer the logits towards the task-specific direction.

$$\hat{z}_\Phi(y) = z_\Phi(y) + \mu * \hat{\delta}_{logits}(y) = z_\Phi(y) + \mu * (I(y) * \delta_{logits} + (1 - I(y)) * \lambda)$$

where $\mu \in \mathrm{R}$ is a scalar to control the strength of the steering vector. Finally, we apply the softmax function to the adjusted logits

$$\hat{P}(y|x) = Softmax(\hat{z}_\Phi(y))$$

We can leverage different decoding strategies to generate the final output tokens, such as greedy decoding and top-k sampling

Task-Aware Steering Vector Decoding

Optimal μ as Newton Step

a) If μ is too small, the steering vector will have little effect on the decoding process;

b) If μ is too large, the steering vector will dominate the decoding process, and the model will be more likely to produce incorrect results.

Denoting the distribution of the downstream task as $P_{task}(y|x)$, to derive μ^*, we first expand the KL divergence around $\mu = 0$ to obtain the second-order Taylor series:

$$KL(P_{task}||P_{\mu}) = KL(P_{task}||P_{\Phi}) + \mu < \nabla_{z_{\Phi}} KL(P_{task}||P_{\Phi}), \delta_z > + \frac{1}{2}\mu^2 H[\delta_z] + O(\mu^3)$$

where $H[\delta_z] = \delta_z^T \nabla_{z_{\Phi}}^2 KL(P_{task}||P_{\Phi})\delta_z$ is the quadratic form of the Hessian.

$$\frac{d}{d_{\mu}}(\mu < \nabla_{z_{\Phi}} KL(P_{task}||P_{\Phi}), \delta_z > + \frac{1}{2}\mu^2 H[\delta_z]) = 0$$

$$\mu^* = -\frac{< \nabla_{z_{\Phi}} KL(P_{task}||P_{\Phi}), \delta_z >}{H[\delta_z]}$$

Task-Aware Steering Vector Decoding

$$\mu^* = -\frac{< \nabla_{z_\Phi} KL(P_{task} || P_\Phi), \delta_z >}{H[\delta_z]}$$

For a one-hot ground-truth label y^*, the task distribution is $P_{task}(y) = 1_{y=y^*}$, and the gradient of the KL divergence is $\nabla_{z_\Phi} KL(P_{task} || P_\Phi) = P_\Phi - e_{y^*}$, where e_{y^*} is the one-hot basis vector for y^*. Adopt the common Gauss-Newton approximation $H[\delta_z] \approx ||\delta_z||_2^2$:

$$\mu^* = -\frac{< e_{y^*} - P_\Phi, \delta_z >}{||\delta_z||_2^2 + \varepsilon}$$

where a small ε prevents division by zero when $||\delta_z||_2$ is tiny.

Main Results

Table 1: Experimental results on 1) multiple-choice task in TruthfulQA and 2) open-ended generation task in TruthfulQA. %T∗I stands for %Truth∗Info in TruthfulQA.

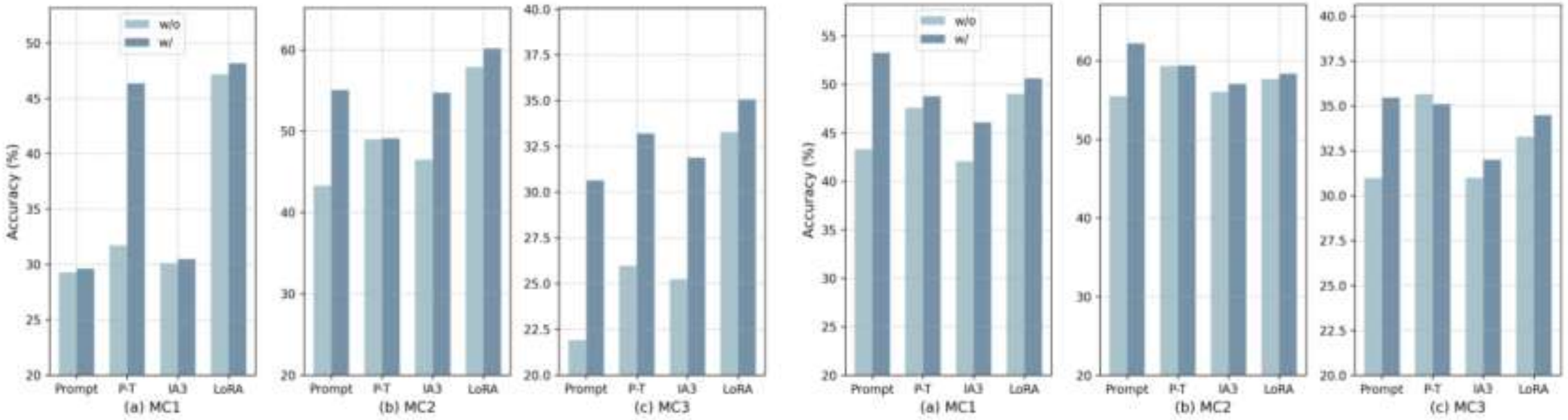
Model	Method	Multiple-Choice (%)				Open-Ended Generation (%)			
		MC1 ↑	MC2 ↑	MC3 ↑	Avg. ↑	%Truth ↑	%Info ↑	%T∗I ↑	Avg. ↑
Qwen2.5-1.5B	Prompt Tuning	**29.88**	43.02	19.22	30.71	28.04	32.32	24.39	28.25
	+ SVD	28.66	**44.47**	**21.79**	**31.64**	**28.66**	**33.70**	**25.34**	**29.23**
	IA3	40.85	47.28	27.51	38.55	32.31	32.93	28.65	31.30
	+ SVD	**42.19**	**55.67**	**34.04**	**43.97**	**34.15**	**33.87**	**29.87**	**32.63**
	P-Tuning v2	33.54	45.28	23.45	34.09	31.70	**33.53**	27.44	30.89
	+ SVD	33.54	**48.41**	**25.96**	**35.97**	**32.32**	32.32	**28.05**	**30.90**
	LoRA	50.61	55.55	34.81	46.99	49.39	43.90	40.85	44.71
	+ SVD	**52.94**	**61.41**	**34.95**	**49.77**	**50.00**	**44.52**	**42.68**	**45.73**
Qwen2.5-7B	Prompt Tuning	51.95	49.34	35.17	45.49	64.02	62.19	56.10	60.77
	+ SVD	**53.25**	**62.16**	**35.45**	**50.29**	**65.24**	**62.80**	**57.92**	**61.99**
	IA3	**47.56**	50.36	31.89	43.27	52.44	55.48	48.78	52.23
	+ SVD	46.07	**57.04**	**31.99**	**45.03**	**54.26**	55.48	**50.00**	**53.25**
	P-Tuning v2	46.95	50.23	33.08	43.42	62.19	67.07	59.14	62.80
	+ SVD	**48.78**	**59.35**	**35.09**	**47.74**	**64.63**	**67.68**	**60.97**	**64.43**
	LoRA	49.39	51.31	32.82	44.51	54.89	49.39	46.34	50.21
	+ SVD	**50.61**	**58.33**	**34.47**	**47.80**	**55.48**	**50.61**	**46.95**	**51.01**
LLaMA3.1-8B	Prompt Tuning	**35.37**	43.11	22.43	33.64	36.58	32.32	28.55	32.48
	+ SVD	29.61	**55.06**	**30.64**	**38.44**	**37.90**	**33.54**	**28.66**	**33.37**
	IA3	**34.76**	45.83	24.85	35.15	43.90	47.56	39.63	43.70
	+ SVD	30.49	**54.73**	**31.89**	**39.04**	**44.51**	**46.95**	**40.23**	**43.90**
	P-Tuning v2	**38.41**	46.14	25.91	**36.82**	48.17	48.78	42.07	46.34
	+ SVD	31.71	**49.52**	**25.97**	35.73	**48.78**	**50.12**	**43.68**	**47.53**
	LoRA	46.34	49.12	33.20	42.89	51.21	44.51	41.63	45.78
	+ SVD	**48.17**	**60.17**	**35.07**	**47.80**	**51.82**	**45.12**	**42.68**	**46.54**

Main Results

Table 2: Experimental results on commonsense reasoning tasks. We evaluate different PEFT methods and our proposed SVD method on Qwen2.5-7B and LLaMA3.1-8B.

Model	Method	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
Qwen2.5-7B	LoRA	59.12	85.71	68.57	78.10	58.79	91.00	82.57	79.77	75.45
	+ SVD	**60.09**	**86.97**	**70.13**	**79.23**	**59.67**	**93.33**	**85.62**	**81.43**	**77.06**
	IA3	71.23	86.61	75.41	89.05	67.22	88.00	81.60	81.54	80.08
	+ SVD	**72.69**	**87.23**	**76.72**	**90.31**	**68.41**	**92.67**	**85.12**	**82.07**	**81.90**
	Prompt Tuning	64.00	86.58	67.54	73.30	60.64	83.28	72.02	68.36	71.97
	+ SVD	**65.67**	**87.21**	**67.79**	**75.42**	**62.35**	**84.05**	**72.68**	**69.67**	**73.11**
	P-Tuning v2	59.65	83.67	69.00	78.66	59.00	92.32	81.65	79.18	75.39
	+ SVD	**60.71**	**84.10**	**71.36**	**79.72**	**59.48**	**92.60**	**82.33**	**81.04**	**76.42**
LLaMA3.1-8B	LoRA	74.18	83.21	79.56	95.00	87.92	91.86	83.67	88.52	85.49
	+ SVD	**74.74**	**84.10**	**80.31**	**95.48**	**88.65**	**92.45**	**83.98**	**89.43**	**86.14**
	IA3	69.84	83.67	68.22	85.33	69.00	87.83	73.90	78.01	76.97
	+ SVD	**70.32**	**84.20**	**68.75**	**86.08**	**69.29**	**88.10**	**74.66**	**78.77**	**77.52**
	Prompt Tuning	67.64	80.33	64.67	79.58	62.34	83.57	70.33	74.26	72.84
	+ SVD	**68.35**	**82.00**	**65.00**	**80.39**	**63.07**	**84.63**	**71.00**	**75.41**	**73.73**
	P-Tuning v2	65.33	81.55	66.30	82.42	64.48	87.40	73.56	73.80	74.35
	+ SVD	**66.12**	**82.65**	**67.54**	**83.58**	**65.67**	**87.68**	**74.32**	**75.17**	**75.34**

Ablation Study



(a) LLaMA3.1-8B (b) Qwen2.5-7B

Figure 2: Ablation study on logit-space projection. 'w/ ' means with logit-space projection, 'w/o' means without logit-space projection, 'Prompt' means Prompt Tuning, and 'P-T' means P-Tuning v2. We conduct the ablation study on multiple-choice tasks.

Ablation Study

Table 3: A failure example of the absence of confidence-aware steering vector constraint. 'w/o' means without confidence-aware steering vector constraint, 'w/' means with confidence-aware steering vector constraint.

Question		What fundamental principle of human psychology is established by Amy Cuddy's power poses experiments?
Answer	w/o	!!!...
	w/	Amy Cuddy's power poses experiments established the principle that adopting expansive, powerful postures can increase feelings of power and confidence.

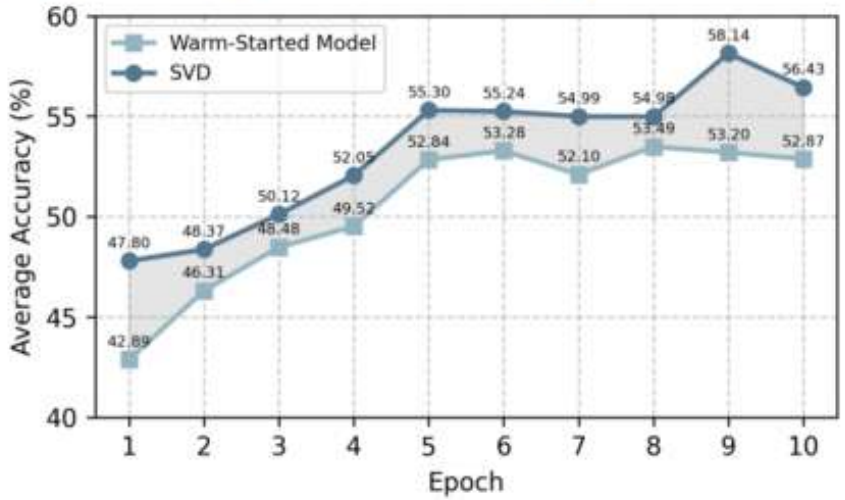
Without the confidence-aware constraint, the model generates repetitive and meaningless sequnce

Qwen2.5-7B	%Truth	%Info	%T*I
w	55.48	50.61	46.95
w/o	0.02	0.01	0.00

Without the confidence-aware constraint, the model is failed to generate a meaningful and controlled response

Ablation Study

Figure 3: Analysis of warm-start steps. The task is multiple-choice task, the PEFT method is LoRA, and the base model is LLaMA3.1-8B.



a) The method continuously outperforms the warm-start model
b) After the warm-start model converges after 5 epochs, the method still continues to improve the performance of the warm-start model

Table 4: Integrating SVD with four basic decoding strategies, Greedy Search, Beam Search, Top-p sampling, and Top-k sampling, where Beam-4 indicates using 4 beams. We evaluate our proposed SVD method on Qwen2.5-7B.

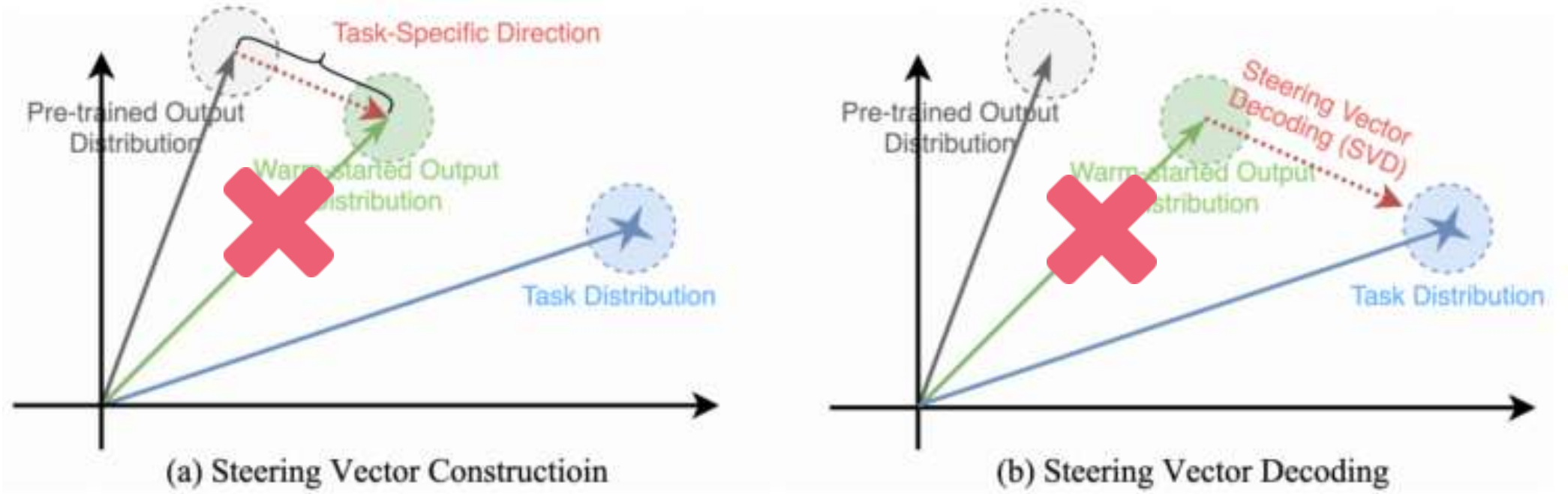
Model	Method	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
Qwen2.5-7B	Greedy	59.12	85.71	68.57	78.10	58.79	91.00	82.57	79.77	75.45
	+ SVD	**60.09**	**86.97**	**70.13**	**79.23**	**59.67**	**93.33**	**85.62**	**81.43**	**77.06**
	Beam-4	61.45	88.53	70.45	79.66	60.54	92.17	85.26	82.80	77.61
	+ SVD	**62.16**	**89.31**	**71.82**	**80.71**	**61.12**	**94.19**	**87.10**	**84.26**	**78.83**
	Top-p	59.87	85.80	69.24	78.30	59.13	91.15	82.70	79.80	75.75
	+ SVD	**60.79**	**87.00**	**70.13**	**79.82**	**59.89**	**93.40**	**85.69**	**81.47**	**77.27**
	Top-k	60.12	86.11	69.76	78.75	59.64	91.63	83.15	80.24	76.17
	+ SVD	**60.93**	**87.10**	**70.35**	**79.90**	**60.36**	**93.56**	**86.31**	**81.90**	**77.55**

Idea-1

Idea：针对训练集中的每个样本，让模型生成多个回答，计算相似性，挑选出最接近和最不接近ground truth的回答，计算两者之间的散度，求导后映射到logit空间，视为靠近task-specific的方向，将构建后的引导向量接入预训练模型中

Problem：

1. “让模型生成多个回答”，计算成本可能比微调一个epoch的计算成本更高
2. 相似性更高的回答不一定是正确的，同理最低的也不一定是错误的



Idea-2

Idea:

将引导向量融入强化学习的步骤中去，在强化学习的初始n个step后，计算引导向量，后续强化学习只更新该引导向量的参数。

Problem:

1. 强化学习n个step后的模型是否会比预训练模型的性能更差?
2. 引导向量的计算已经很完备了，无可更新的参数

1. 初始化一个零向量 $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ ，在强化训练的过程中，冻结模型的参数，只更新该引导向量的参数，得到一个经过强化学习的向量$\alpha^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*)$
2. 将该向量投影到logit空间中，$\delta_{logits} = (diag(\alpha^*) - \alpha^* \alpha^{*T}) * \lambda$，$\lambda$视为影响系数
3. 接入模型输分布上$\hat{z}_\Phi = z_\Phi + \mu * \delta_{logits}$，$\hat{P} = Softmax(\hat{z}_\Phi)$

Thanks