Self-Concordant Preference Learning from Noisy Labels

Shiv Shankar [1] **Madalina Fiterau** [1]

Abstract

Preference learning is an integral part of the training process for a large language model (LLM) to serve user applications. While this alignment is usually done via offline learning from annotated feedback, there is inherent noise in obtaining such data, and most current methods are sensitive to such noise. In this work, we propose a novel approach to use such noisy labels based on concordant losses. Our proposed method is based on learning the optimal model under an adversarial labeller. Experiments show that our proposal is more effective than common algorithms for various levels of noise.

1. Introduction

The development of large language models has significantly revolutionized the modern AI field (Hadi et al., 2023; Fan et al., 2024). A major contributor towards the success of these models is Reinforcement Learning from Human Feedback (RLHF) based preference learning (Achiam et al., 2023; Ouyang et al., 2022). A standard RLHF based alignment pipeline involves: 1) Supervised Fine Tuning (SFT) of a pretrained LLM, 2) Learning Reward Model from human annotated data for guiding the LLM and 3) Reinforcement Learning (RL) based optimization using the reward model.

This standard alignment approach treats human-supplied annotation as if they were clean and helpful signals. But this data annotation process is, by its very nature, error-prone. In practice, annotators disagree, change their minds, and occasionally behave adversarially. Furthermore, for many areas which might require expert annotators, it can be hard to get enough high-quality 'gold' preference data. The resulting "silver" preference data are inevitably biased and noisy. Despite these realities, most alignment pipelines simply plug such feedback into a reward model and hope for the best, silently inheriting the fragile assumptions of standard supervised learning.

[1]University of Massachusetts. Correspondence to: <sshankar@cs.umass.edu>.

ICML 2025 Workshop on Models of Human Feedback for AI Alignment., Vancouver, Canada.

Contribution This paper tackles the problem: How can we reliably learn models when the human feedback can be wrong or corrupted? We propose a self-concordance (Bach, 2010; Pleiss et al., 2020) based method for training from such partially noisy data. First, we look at a prediction function that is robust to adversarial labelling. We propose to learn the robust model by distilling (Buciluǎ et al., 2006; Hinton, 2015; Gou et al., 2021) from the reward function learned in the aforementioned adversarially robust procedure. For efficient learning, we turn to a tractable approximation of the robust prediction for logistic learning (Bach, 2010) using influence functions. Experiments show the efficacy of our proposal.

2. Background and Related Work

The language model is considered as a policy function π that observes a prompt x and produces a textual response a by sampling from a distribution $y \sim \pi(\cdot \mid x)$. We are given a dataset $\mathcal{D}_{\text{pref}} = (x, y)$ consisting of prompts x and a pair of labelled response $y = (y^{+}, y^{-})$ where y^{+} represents the preferred response, and y^{-} represents the negative response.

RLHF (Christiano et al., 2017; Ye et al., 2024; Ouyang et al., 2022) deals with the problem of aligning a language model, using $\mathcal{D}_{\text{pref}} = \{(x, y)\}$. Given the context/prompt x, a pair of outputs are sampled from $\pi_{\mathsf{ref}}(\cdot \mid x)$ and then arranged as per preference function (typically implicitly given by human annotation). RLHF methods (Christiano et al., 2017; Ouyang et al., 2022) seek to obtain a policy $\hat{\pi}$ that is aligned with the preference data. This is usually done by first estimating a reward function $\hat{r}$ from $\mathcal{D}_{\text{pref}}$, by fitting a *Bradley-Terry* model (Bradley and Terry, 1952) via maximum likelihood.

The estimated reward function $\hat{r}$ is then used to train the LLM by RL-based methods. The model $\hat{\pi}$ is regularized to stay close to a reference policy π_{ref} (typically the SFT model), giving the following objective:

$$\hat{\pi} = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\pi}\left[\hat{r}(x, y) - \beta \log \frac{\pi(a \mid x)}{\pi_{\mathsf{ref}}(a \mid x)}\right].$$

The optima for the above model is given by the energy model $\hat{\pi} \propto \pi_{\mathsf{ref}} \exp(\hat{r}/\beta)$ (Ziebart et al., 2008). Using this insight, the DPO method (Rafailov et al., 2024) relies on optimizing

π directly via plugging the corresponding implied reward function in the MLE objective:

$$\max \sum_{(x,y^+,y^-)\in\mathcal{D}_{\text{pref}}} \log\sigma\left(\beta\frac{\pi_\theta(y^+\mid x)}{\pi_{\text{ref}}(y^+\mid x)} - \frac{\pi_\theta(y^-\mid x)}{\pi_{\text{ref}}(y^-\mid x)}\right), \quad (1)$$

where we have used a parametric model π_θ parameterized by θ. We overload the notation and use the same parameter θ for the reward r. Many methods (Ethayarajh et al., 2024; Azar et al., 2023) build upon this insight and instead of training the reward model $\hat{r}$, directly try to optimize some version of Equation 1 by changing the objective from the logistic loss to a broader family of loss functions.

Generalization and Robust Learning A common problem in tuning LMs using this methodology is reward hacking (Huang et al., 2024; Eisenstein et al., 2023). Specifically, the RL method can tune the LM to fit any idiosyncrasies or biases in the reward model (Rita et al., 2024) and fails to generalize outside the preference data. Some recent proposals (Coste et al., 2024; Cheng et al., 2023) to overcome this problem involve adversarial optimization. Other proposals include regularization with stronger divergence penalties (Huang et al., 2024), ensemble models (Eisenstein et al., 2023) and uncertainty-based learning (Zhai et al., 2023). Robust preference learning solves an extreme form of generalization problem in that the reward function needs to be robust not only to the finite preference data but also to mislabeled data. Most methods of robust learning rely on filtering out mislabeled examples (Wang et al., 2020), some form of Bayesian learning(Yang et al., 2024), or down-weighting examples (Liang et al., 2024).

Our approach is most connected to the adversarial learning methodology of Cheng et al. (2023). This family of methods, while principled, can be computationally unstable due to using an explicit adversarial learning procedure. Furthermore, most of these methods directly rely on an explicit reward function, which can still lead to overoptimization (Zhang et al., 2024). On the other hand, down-weighting examples, while intuitive, lack a principled justification.

3. Proposed Method

One can consider Equation 1 as a specifically parameterized binary classifier on the input $z = (x, y)$ where $f_\theta(z) = r_\theta(x, y^+) - r_\theta(x, y^-)$. Furthermore, since we are combining both y^+, y^- under y, we define an auxiliary variable y_z, which is 1 iff y^+ is preferred over y^- and 0 otherwise. For most of this section, we will use this notation instead. Under this change, the DPO/MLE objective is effectively a regression problem :

$$\begin{aligned}\mathcal{L}(f) &= \mathbb{E}\left[y_z \log(\sigma(f(z))) + (1-y_z)\log(\sigma(1-f(z)))\right] \\ &= \sum_{z\sim\mathcal{D}_{\text{pref}}} l(f(z), y_z) \quad (2)\end{aligned}$$

with l being the logistic loss between label y_z and model prediction $f(z)$ and we supress dependence on θ for notational convenience.

Robust Learning via MinMax Model Our idea of learning under mislabeled labels stems from the ideas on concordant losses (Ostrovskii and Bach, 2021; Bach, 2010; Mourtada and Gaïffas, 2022) Let us consider a single observation z distinct from observations in $\mathcal{D}_{\text{pref}}$. For this additional sample, we consider the following sample minimax prediction:

$$\hat{f}^{z,y} = \operatorname{argmin}\left\{\mathcal{L}(f) + l(f(z), y)\right\} \quad (3)$$

$$\hat{y}(z) = \operatorname*{argmin}_{y'} \max_{y} \left\{l(y', y) - l(\hat{f}^{z,y}(z), y)\right\} \quad (4)$$

where $\hat{f}^{z,y}$ is a model trained with data and additional sample (z, y), and $\hat{f}^{z,y}(z)$ is the same model applied on input z. Note that y, y' here are arbitrary variables in the output space of the model and different from y_z, which is the label of z in $\mathcal{D}_{\text{pref}}$.

The intuition behind the above is as follows: Suppose we add an additional sample z to the training data, labeled with y. Then, Equation 3 fits the model $\hat{f}^{z,y}$ on this updated dataset. The loss incurred by this model on the specific sample (z, y) is given by $l(\hat{f}^{z,y}(z), y)$. Alternatively, if instead of using $\hat{f}^{z,y}$ we simply predict y', the resulting loss is $l(y', y)$. Since $l(\hat{f}^{z,y}(z), y)$ represents the irreducible loss from fitting the model, the best we can aim to optimize is the excess loss, defined as $l(y', y) - l(\hat{f}^{z,y}(z), y)$.

Remark. *Note that we consider the adversarial label at an individual instance level. This is different from poisoning which often aims to change prediction of a certain target input. It also does not consider interaction between mislabelled samples, where the adversary mislabels an entire set of points together to maximize regret.*

Now consider a scenario where an adversary selects the label y for the observation z with the goal of maximizing our risk. This justifies the maximization over y. In contrast, we aim to minimize the loss by selecting an optimal prediction y', motivating the subsequent minimization over y'.

Equation 4 is, in general, hard to solve, however for logistic regression, one has the following closed-form result (Mourtada and Gaïffas, 2022):

$$\hat{y}(z) = \frac{\sigma(\hat{f}^{z,1})}{\sigma(\hat{f}^{z,1}) + 1 - \sigma(\hat{f}^{z,0})} \tag{5}$$

The above expression has the following semantics: train the model with z being given each label $0, 1$ once. On $\mathcal{D}_{\text{pref}} + (z, 1)$, we get $\hat{f}^{z,1}$ and with $\mathcal{D}_{\text{pref}} + (z, 0)$ we have $\hat{f}^{z,0}$. We then normalize the predicted scores to compute the output probability.

The semantics of this expression are as follows: we train the model twice, once with the sample z labeled as 1, and once as 0. Specifically, training on $\mathcal{D}_{\text{pref}} \cup (z, 1)$ yields $\hat{f}^{z,1}$, while training on $\mathcal{D}_{\text{pref}} \cup (z, 0)$ yields $\hat{f}^{z,0}$. The predicted scores from both models are then normalized to produce the estimated probability $\hat{y}(z)$.

Consider the case where z is close (in feature space) to many positive examples (samples with label 1). In this case, $\hat{f}^{z,1}(z)$ is close to the prediction from a model trained without z, and is typically close to 1. Conversely, when training on $\mathcal{D}_{\text{pref}} \cup (z, 0)$, the model incurs a high loss on z and must shift decision boundaries to reduce $\hat{f}^{z,0}(z)$ toward 0. However, doing so requires contradicting many nearby positive samples, so the model may only reduce the score moderately (e.g., to 0.6). This results in $\hat{y}(z) \approx 0.71$: high, but not close to 1.

This aligns with our intuition - if most of the dataset is correctly labeled, and z is strongly supported by positive examples, we can reasonably (though not with full certainty) assign label $y = 1$ to z. On the other hand, if z has no support in the data, f can fit both data equally well. Then both $\hat{f}^{z,1}(z)$ and $\hat{f}^{z,0}(z)$ may be close (e.g., both near 1), yielding $\hat{y}(z) \approx 0.5$—an ambiguous prediction. Thus the model naturally incorporates the idea of support from other samples support.

We propose to learn a new model by using the above technique on a standard MLE-trained reward model and then distilling from the new predictions. Specifically, we propose to first train a model, get the new prediction of labels y_z^0 for every $z \in \mathcal{D}_{\text{pref}}$, create a new dataset $\mathcal{D}^1_{\text{pref}} = (z, y_z^0) \forall z \in \mathcal{D}_{\text{pref}}$ and then use it as a distillation target for training the new model. However, doing so requires us to be able to train a model for each observation in the dataset $\mathcal{D}_{\text{pref}}$, which is computationally intractable. To alleviate this problem, we turn to influence functions.

Influence Functions Influence functions (Cook and Weisberg, 1980; Johnson, 1985) characterise how a model's loss or predictions depend on training data. Closest to the proposal in this work, influence functions have also been used for detecting outliers and corrupted samples (Dau et al., 2022; Wang et al., 2020).

One way to characterize influence functions is to consider placing ϵ additional weight on sample z_i and taking $\lim_{\epsilon \to 0}$. For the parametric model, f_θ trained by minimizing $\mathcal{L}(f)$ one can show the 'influence' of an observation z_i on the prediction of another sample z_j as (Koh and Liang, 2017):

$$\hat{I}(z_i, z_j) = -\nabla_\theta f_\theta(z_j)^T H_\theta^{-1} \nabla_\theta l(z_i, \theta) \tag{6}$$

where we have overloaded the notation by defining $l(z_i, \theta) = l(f_\theta(z_i), y_{z_i})$ as the loss of the model f_θ on the sample z_i, y_{z_i}. H_θ is the Hessian matrix of the loss, and $\nabla_\theta l(z_i, \theta)$ is the gradient of the loss, and $\nabla_\theta f$ is the gradient of the prediction. The negative sign is due to f being learnt by minimization of l.

Using the influence function one can compute $\hat{f}^{z_i,y}$ for the observation z_i and label y as :

$$\begin{aligned} \hat{f}^{z_i,y} &= f_\theta(z_i) - \alpha \hat{I}(z_i, z_i) + \alpha \hat{I}(z_i, (z_i, y)) \\ &= f_\theta(z_i) + \alpha \nabla_\theta f_\theta(z)^T H_\theta^{-1} [-\nabla_\theta l(f_\theta(z_i), y) \\ &\qquad + \nabla_\theta l(f_\theta(z_i), y_{z_i})]. \end{aligned} \tag{7}$$

The above equation can be interpreted as removing the influence of the observation z_i, y_{z_i} and then adding the influence of observation z_i, y, α is the step size used for update and is a hyperparameter. With $f^{z_i,y}$ computed, we can get $\hat{y}(z)$ from Equation 5. We then distill it into a DPO model by optimizing Equation 2 treating it $\hat{y}(z)$ as the smoothened label y_z.

Efficient Computation One issue in utilizing the above expression directly is that it requires computing and inverting the Hessian, which is intractable for LLMs. Fortunately, there has been a lot of research towards fast and scalable estimation of influences. A common approximation used is a diagonal Hessian (Klochkov and Liu, 2024; Thimonier et al., 2022). Under such a diagonal approximation the influence term is an inner product between gradients. Such an inner product can be computed in one backward pass using the 'ghost clipping' technique (Bu et al., 2023). The key idea behind this is mentioned here and we refer the readers to Bu et al. (2023); Wang et al. (2024) for greater details.

Consider a single linear layer which maps features $\mathbf{x}_1$ to $\mathbf{x}_2$

$$\mathbf{x}_2 = W \mathbf{x}_1, \qquad W \in \mathbb{R}^{d_1 \times d_2},\ \mathbf{x}_1 \in \mathbb{R}^{d_1},\ \mathbf{x}_2 \in \mathbb{R}^{d_2}.$$

For a training example i with loss ℓ^i, the gradient with respect to W is

$$\nabla_W \ell_i \ = \ \frac{\partial \ell_i}{\partial \mathbf{x}_2^i} \frac{\partial \mathbf{x}_2^i}{\partial W} \ = \ g_i \, \mathbf{x}_1^{i\top}, \tag{8}$$

where $g_i = \partial \ell / \partial \mathbf{x}_2^i \in \mathbb{R}^{d_2}$ is the back-propagated signal. Equation (8) shows that every per-sample gradient is a *rank-1 outer product*.

Let $\mathbf{x}^j$ denote another example. The inner product of per-parameter gradients becomes

$$\langle \nabla_W \ell_j, \nabla_W \ell_i \rangle = \langle g_j \mathbf{x}_1^{j\top}, g_i \mathbf{x}_1^{i\top} \rangle = \underbrace{g^{j\top} g_i}_{\text{length } d_2} \underbrace{\mathbf{x}_1^{j\top} \mathbf{x}_1^i}_{\text{length } d_1}. \quad (9)$$

Thus the potentially expensive $d_1 d_2$-dimensional inner product factorises into *two* cheap dot products of lengths d_2 and d_1, respectively. Further the gradients with respect to each layer embedding itself gets computed for each element in the backward pass. Thus we can compute the inner product for elements in a batch with only one backward pass.

4. Experiments

Tasks Following earlier works, we experiment with the Alpaca Comparison Benchmark (Peng et al., 2023), a dialogue task and a forum summarization dataset Reddit TLDR (Völske et al., 2017). The Alpaca Comparison dataset contains queries from the benchmark Alpaca dataset (Taori et al., 2023). We follow the work of Peng et al. (2023), which generates responses using smaller LLMs and then scores each response for the prompt using GPT-4. TLDR is a dataset where the prompt consists of the forum content, and the task is to summarize the forum. We follow the procedure of Rafailov et al. (2024) for this dataset using human preference annotation as done by Stiennon et al. (2020).

We experiment with different noise settings, where we, at random, select and flip the preference labels of p fraction of the data.

Models and Baselines We experiment with Mistral-7B (Jiang et al., 2023) and LLama-2-7B (Touvron et al., 2023) for learning preferences. For preference learning algorithms we use DPO (Rafailov et al., 2024), IPO (Azar et al., 2023), and two other approaches (rDPO (Chowdhury et al., 2024) and cDPO (Mitchell, 2023)) which induce robustness in DPO by using label smoothening on the MLE loss.

Evaluation For Alpaca data, we can evaluate the models using the AlpacaEval benchmark(Li et al., 2023), by comparing their outputs with those of GPT3 (text-davinci) (Achiam et al., 2023). For TLDR, we compare their outputs against the SFT response and, following standard procedure in literature (Liang et al., 2024; Zhang et al., 2024), use GPT-4 for scoring the answers. The prompt used for scoring TLDR is a slight variation of the prompt for Alpaca, and can be found in Liang et al. (2024).

Results We present the results from the described experiment in Tables 1 and 2 for the Mistral and Llama models, respectively. Firstly, one can observe that, as one might expect, all models deteriorate in their performance as the label noise increases. This is natural as more label noise means learning preference is harder. More importantly, we can see that our model shows the lowest deterioration compared to other methods, highlighting the robustness of the overall approach. We also see mild improvements even at 0% noise. This is because of the smoothening effect of the minimax procedure, which prevents the implicit reward scores from growing very high and acts as implicit regularization. This effect is stronger when we add noise to the data; these are the cases when our model significantly outperforms other methods. The results are consistent across both Mistral and Llama models.

Table 1: Win Rates of different methods using Mistral7-B compared to the SFT baseline with different proportions of corrupted preference labels.

Method	Alpaca				Reddit TLDR			
	0%	5%	10%	20%	0%	5%	10%	20%
DPO	69.5	69.0	68.6	68.5	62.9	59.5	57.2	56.4
IPO	68.8	66.6	64.9	64.1	62.1	59.5	57.6	56.2
rDPO	68.6	68.5	68.3	68.1	62.2	61.2	58.4	55.7
cDPO	65.1	64.6	63.2	61.8	59.8	59.3	57.9	56.6
Ours	70.3	**70.2**	**70.1**	**68.9**	63.1	**63.0**	**62.7**	**59.1**

Table 2: Win Rates of different methods using Llama-2-7B compared to the SFT baseline with different proportions of corrupted preferences.

Method	Alpaca				Reddit TLDR			
	0%	5%	10%	20%	0%	5%	10%	20%
DPO	51.5	48.8	46.7	46.0	57.2	50.2	44.8	43.0
IPO	51.8	50.8	50.0	49.7	54.7	52.5	50.8	50.2
rDPO	49.8	49.6	49.0	48.4	54.2	53.5	51.7	49.9
cDPO	52.1	51.7	50.6	49.5	52.1	51.4	49.4	47.6
Ours	52.2	**52.1**	**52.0**	**50.9**	56.9	**56.8**	**56.5**	**53.3**

5. Conclusion

Preference optimization of LLMs under noisy annotation is an important research problem. Most existing methods rely on some version of label smoothing or gradient reweighing. We propose an alternative based on adversarially robust prediction and influence functions. Specifically, we show how one can use per-sample minimax labelling to predict robust labels even under adversarial noise. The resulting method relies on training the model with and without the corresponding sample, which is computationally intensive. This is approximated using the recently proposed KFAC-based influence estimation. Experiments show the effectiveness of our proposed methodology.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. (2023). A general theoretical paradigm to understand learning from human

preferences.

Bach, F. (2010). Self-concordant analysis for logistic regression.

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39.

Bu, Z., Wang, Y.-X., Zha, S., and Karypis, G. (2023). Differentially private optimization on large model at small cost. In *International Conference on Machine Learning*, pages 3192–3218. PMLR.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Cheng, P., Yang, Y., Li, J., Dai, Y., and Du, N. (2023). Adversarial preference optimization. *arXiv preprint arXiv:2311.08045*.

Chowdhury, S. R., Kini, A., and Natarajan, N. (2024). Provably Robust DPO: Aligning Language Models with Noisy Feedback. *arXiv e-prints*.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences.

Cook, R. D. and Weisberg, S. (1980). Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508.

Coste, T., Anwar, U., Kirk, R., and Krueger, D. (2024). Reward model ensembles help mitigate overoptimization.

Dau, A. T., Bui, N. D., Nguyen-Duc, T., and Thanh-Tung, H. (2022). Towards using data-influence methods to detect noisy samples in source code corpora. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–3.

Eisenstein, J., Nagpal, C., Agarwal, A., Beirami, A., D'Amour, A., Dvijotham, D., Fisch, A., Ramachandran, D., Shaw, P., and Berant, J. (2023). Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. (2024). Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. (2024). A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Hadi, M. U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M. B., Akhtar, N., Wu, J., Mirjalili, S., et al. (2023). A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

Hinton, G. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Huang, A., Zhan, W., Xie, T., Lee, J. D., Sun, W., Krishnamurthy, A., and Foster, D. J. (2024). Correcting the mythos of kl-regularization: Direct alignment without overoptimization via chi-squared preference optimization.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Singh Chaplot, D., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Renard Lavaud, L., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., and El Sayed, W. (2023). Mistral 7B. *arXiv e-prints*.

Johnson, W. (1985). Influence measures for logistic regression: Another point of view. *Biometrika*, 72(1):59–65.

Klochkov, Y. and Liu, Y. (2024). Revisiting inverse hessian vector products for calculating influence functions.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Li, X., Zhang, T., Dubois, Y., and Taori, R. (2023). Alpacaeval: An automatic evaluator of instruction-following models.

Liang, X., Chen, C., Wang, J., Wu, Y., Fu, Z., Shi, Z., Wu, F., and Ye, J. (2024). Robust preference optimization with provable noise tolerance for llms. *arXiv preprint arXiv:2404.04102*.

Mitchell, E. (2023). A note on dpo with noisy preferences and relationship to ipo.

Mourtada, J. and Gaïffas, S. (2022). An improper estimator with optimal excess risk in misspecified density estimation and logistic regression. *Journal of Machine Learning Research*, 23(31):1–49.

Ostrovskii, D. M. and Bach, F. (2021). Finite-sample analysis of m-estimators using self-concordance.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and

Lowe, R. (2022). Training language models to follow instructions with human feedback.

Peng, B., Li, C., He, P., Galley, M., and Gao, J. (2023). Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Pleiss, G., Zhang, T., Elenberg, E., and Weinberger, K. Q. (2020). Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. (2024). Direct preference optimization: your language model is secretly a reward model. NIPS '23.

Rita, M., Strub, F., Chaabouni, R., Michel, P., Dupoux, E., and Pietquin, O. (2024). Countering reward over-optimization in llm with demonstration-guided reinforcement learning.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. (2020). Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. (2023). Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Thimonier, H., Popineau, F., Rimmel, A., Doan, B.-L., and Daniel, F. (2022). Tracinad: Measuring influence for anomaly detection. In *2022 International joint conference on neural networks (IJCNN)*, pages 1–6. IEEE.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Singh Koura, P., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv e-prints*.

Völske, M., Potthast, M., Syed, S., and Stein, B. (2017). Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63.

Wang, J. T., Wu, T., Song, D., Mittal, P., and Jia, R. (2024). Greats: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems*, 37:131197–131223.

Wang, Z., Zhu, H., Dong, Z., He, X., and Huang, S.-L. (2020). Less is better: Unweighted data subsampling via influence function. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6340–6347.

Yang, A. X., Robeyns, M., Coste, T., Wang, J., Bou-Ammar, H., and Aitchison, L. (2024). Bayesian reward models for llm alignment. *arXiv preprint arXiv:2402.13210*.

Ye, C., Xiong, W., Zhang, Y., and Zhang, T. (2024). Online iterative rl from human feedback with general preference model.

Zhai, Y., Zhang, H., Lei, Y., Yu, Y., Xu, K., Feng, D., Ding, B., and Wang, H. (2023). Uncertainty-penalized reinforcement learning from human feedback with diverse reward lora ensembles. *arXiv preprint arXiv:2401.00243*.

Zhang, X., Ton, J.-F., Shen, W., Wang, H., and Liu, Y. (2024). Overcoming reward overoptimization via adversarial policy optimization with lightweight uncertainty estimation. *arXiv preprint arXiv:2403.05171*.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. (2008). Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA.